NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

DEVELOPMENT OF A NONLINEAR 6-DEGREE OF FREEDOM MINIATURE ROTARY-WING UNMANNED AERIAL VEHICLE SOFTWARE MODEL AND PID FLIGHT PATH CONTROLLER USING MATHWORKS SIMULINK SIMULATION ENVIRONMENT

by

Ricardo E. Miranda

September 2009

Thesis Advisor:	Robert G. Hutchins
Co-Advisors:	Vladimir Dobrokhodov
	Ioannis Kitsios

Approved for public release; distribution is unlimited

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** September 2009	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: Development of a Nonlinear 6-Degree of Freedom Miniature Rotary-wing Unmanned Aerial Vehicle Software Model and PID Flight Path Controller Using MathWorks Simulink Simulation Environment			**5. FUNDING NUMBERS**
6. AUTHOR(S): Ricardo E. Miranda			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

This paper describes the development of a 6-degree of freedom (6-DOF), nonlinear, miniature rotary-wing unmanned aerial vehicle (RW UAV) simulation environment using MathWorks Simulink simulation software. In addition to the modeling process, this research also conducts flight-path controller design using Proportional Integral Derivative (PID) control. This model's development is motivated by the desire to enable a rapid prototyping platform for design and implementation of various flight control techniques with further seamless transition to the hardware in the loop (HIL) and flight-testing. The T-Rex Align 600 remote controlled helicopter with COTS autopilot was chosen as a prototype rotary UAV platform.

The development of the nonlinear simulation model is implemented starting with extensive literature review of helicopter aerodynamics and flight dynamics theory and applying the mathematical models of the helicopter components to generate helicopter inertial frame motion simulations from pilot commands. The primary helicopter components modeled in this thesis include the helicopter main rotor inflow, thrust, flapping dynamics, as well as the tail rotor inflow and thrust responses. The inertial frame motions are animated using the Flight Gear Version 0.9.8 software.

After obtaining simulations with verifiable results, the nonlinear model is linearized about the hovering flight condition and a linear model is extracted. Lastly, the PID controller is designed and flight path software in the loop (SIL) test results are presented and explained.

14. SUBJECT TERMS: Hardware in the Loop (HIL), Software in the Loop (SIL) Simulation Environment, 6-Degree of Freedom (6-DOF) Rotary-wing Unmanned Aerial Vehicle (RW UAV) model , PID Flight Path Controllers			**15. NUMBER OF PAGES** 135
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

DEVELOPMENT OF A NONLINEAR 6-DEGREE OF FREEDOM MINIATURE ROTARY-WING UNMANNED AERIAL VEHICLE SOFTWARE MODEL AND PID FLIGHT PATH CONTROLLER USING MATHWORKS SIMULINK SIMULATION ENVIRONMENT

Ricardo E. Miranda
Captain, United States Marine Corps
B.S., University of Florida, 2002

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 2009

Author: Ricardo E. Miranda

Approved by: Robert G. Hutchins
Thesis Advisor

Vladimir Dobrokhodov
Co-Advisor

Ioannis Kitsios
Co-Advisor

Jeffrey B. Knorr
Chairman, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This paper describes the development of a 6-degree of freedom (6-DOF), nonlinear, miniature rotary-wing unmanned aerial vehicle (RW UAV) simulation environment using MathWorks Simulink simulation software. In addition to the modeling process, this research also conducts flight-path controller design using Proportional-Derivative (PD) control techniques. This model's development is motivated by the desire to enable a rapid prototyping platform for design and implementation of various flight control techniques with further seamless transition to the hardware in the loop (HIL) and flight-testing. The T-Rex Align 600 remote controlled helicopter with COTS autopilot was chosen as a prototype rotary UAV platform.

The development of the nonlinear simulation model is implemented starting with extensive literature review of helicopter aerodynamics and flight dynamics theory and applying the mathematical models of the helicopter components to generate helicopter inertial frame motion simulations from operator commands. The primary helicopter components modeled in this thesis include the helicopter main rotor inflow, thrust, flapping dynamics, as well as the tail rotor inflow and thrust responses. The inertial frame motions are animated using the Flight Gear Version 0.9.8 software.

After obtaining simulations with verifiable results, the nonlinear model is linearized about the hovering flight condition and a linear model is extracted. Lastly, the PD controller is designed and flight path software in the loop (SIL) test results are presented and explained. The SIL tests are conducted for autonomous flight along specified rectangular and figure-8 flight paths.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

The use of rotary-wing unmanned aerial vehicles (RW UAV) brings an unparalleled advantage in defense applications. Unlike its fixed-wing counterparts, rotary-wing platforms provide much more flexibility and maneuverability in the joint battle space. The RW UAV is more flexible because it does not require a cleared runway or additional airspace for landing or takeoff maneuvers. Because of its vertical takeoff, landing, and flight capabilities, the RW UAV makes an excellent platform for low altitude aerial reconnaissance of urban terrain.

Harnessing this potential demands the development of autopilot control algorithms that are testable and robust. Autopilot and RW UAV hardware is expensive, and no current testing software can simulate the flight dynamics of the Mechanical and Astronautical Engineering (MAE) Department's miniature RW platform, the T-Rex Align 600. The development of a simulation environment is a step closer towards being able to design and test control algorithms while lowering the risk of breaking expensive hardware components during flight tests. The intent of the thesis research conducted herein is to enable and facilitate the development of a miniature RW simulation environment. It is hoped that the results provided here will spawn further interest and development efforts that will lead to a more accurate and reliable control test bed.

The first stage in the development of the simulation environment is the modeling of the 6-DOF RW dynamics of the miniature helicopter. This is done by conducting extensive literature review of the authoritative helicopter aerodynamics and flight dynamics theory, and applying the mathematical models of the helicopter components to generate helicopter inertial frame motion simulations from control inputs. The primary helicopter components modeled in this thesis include the helicopter main rotor inflow, thrust, flapping dynamics, as well as the tail rotor inflow and thrust responses. The RW model responses are then qualitatively compared to the data and results of [1] through [7]. The software 6-DOF model was created using MathWorks Simulink simulation software. The inertial frame motions are animated using the Flight Gear Version 0.9.8 software.

The second stage in the development of the simulation environment is the linearization of the nonlinear model. Controller design focuses on the hovering flight regime and linearization helps to uncover the state boundaries for which the controller will be useful. The last stage of the development of the simulation environment was the proportional-derivative (PD) controller tuning and implementation on the nonlinear model. The PD controller performance was tested by analyzing flight path tracking performance.

The complete 6-DOF model performs well in modeling the dynamics of a miniature RW helicopter as compared to the predictions and flight test data provided by [1] through [7]. In addition, analysis of the flight path tracking performance leads to the conclusion that the derived PD gains provide satisfactory control of the model. Maximum errors in position and yaw angle response are expected and are within tolerance.

It is hoped that the research and results from this effort will lead to further development and refinement of the simulation environment that has been developed up to this point.

NOMENCLATURE

Latin Variables

C_T	Coefficient of thrust $C_T = \frac{T}{\rho(\Omega R)^2 \pi R^2}$	[•]
I_{xx}, I_{yy}, I_{zz}	Helicopter Principal Moments of Inertia [.18 .34 .28]	[kg m^2]
K_β	MR Hub Stiffness	[Nm/rad]
L, M, N	Helicopter Roll, Pitch, and Yaw moments	[N m]
L_b, M_a	Roll Moment and Pitch Moment Derivative	[Nm/rad]
h_{tr}	Tail rotor moment arm. Vertical length from tail rotor hub to helicopter c.g. line of sight	.08[m]
l_{tr}	Tail rotor moment arm. Length from helicopter c.g. to tail rotor hub	.91[m]
$\dot{m}$	Mass flow rate of air through rotor	[kg/sec]
N	Number of blades	
p, q, r	Helicopter body roll, pitch, and yaw rates, respectively	[rad/sec]
T	Main rotor thrust	[N]
T_{tr}	Tail rotor thrust	[N]
u, v, w	Helicopter airspeed in the x, y, and z directions, respectively	[m/s]
u_{tr}, v_{tr}, w_{tr}	Tail rotor hub airspeed in the x, y, and z directions, respectively	[m/s]
v_i	Normal component of rotor induced velocity	[m/s]
$v_{i\infty}$	Induced flow of air through rotor, far downstream of rotor	[m/s]
V, V_∞	Helicopter airspeed (magnitude)	[m/s]
V_n	Velocity of air normal to rotor	[m/s]
X, Y, Z	Helicopter forces in the X, Y and Z directions	[N]

Greek Variables

α	Angle of attack. Angle between horizon and main rotor tip path Plane	[rad]
$\beta_{lon},\ \beta_{lat}$	Longitudinal and Lateral Main Rotor Flap Angles	[rad]
ρ	Air density (1.275 kg/m^3 at sea level)	[kg/m^3]
$\lambda_i,\ \lambda_{mr}$	Inflow ratio for main rotor	[•]
μ	Advance ratio. Ratio between helicopter translational velocity and main rotor tip speed	[•]
μ_z	Advance ratio along the z axis. Ratio of vertical velocity to main rotor tip speed	[•]
u_{col}	Collective lever input ($0 \le u_{col} \le 1$)	[•]
u_{lat}	Lateral stick input (left-right $-1 \le u_{lat} \le 1$)	[•]
u_{lon}	Longitudinal stick input (forward-aft $-1 \le u_{lon} \le 1$)	[•]
u_{ped}	Rudder pedal input (left-right $-1 \le u_{ped} \le 1$)	[•]
Ω	Main Rotor rotation rate	167[rad/sec]
Ω_{tr}	Tail Rotor rotation rate	778.2 [rad/sec]
θ_o	Collective pitch-angle of main rotor blade	[rad]
θ_{lat}	Lateral cyclic-pitch angle Angle of main rotor blade when blade is aligned with the $+y$-axis	[rad]
θ_{lon}	Longitudinal cyclic-pitch angle. Angle of main rotor blade when blade is aligned with the $-x$-axis	[rad]
δ_{rud}	Collective pitch-angle of tail rotor blade	[rad]
$\phi,\ \theta,\ \psi$	Euler orientation angles	[rad]
τ_f	Main Rotor Time Constant	[sec]

Constants

a	Main rotor lift curve slope	5.5[rad^{-1}]
a_{tr}	Tail rotor lift curve slope	5.0[rad^{-1}]
c	Rotor blade chord	[m]
A_d	Main rotor disk area	1.887[m^2]
g	Gravitational acceleration	9.82[m/s^2]
m	Mass of helicopter	8.2[kg]
R	Main Rotor radius	.775[m]
R_{tr}	Tail Rotor radius	.13[m]
s	Rotor Solidity $Nc/(\pi R)$	[•]

Abbreviations

MR	Main Rotor
fus	Fuselage
tr	Tail Rotor

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGEMENTS

I owe a debt of gratitude to Professor Robert G. Hutchins, of the Electrical Engineering Department, for his willingness to adopt me as his thesis student and for the unwavering support he provided throughout the research process. I also want to thank my Co-Advisors Professor Vladimir Dobrokhodov and Dr. Ioannis Kitsios, of the Mechanical and Astronautical Engineering (MAE) Department, for allowing me to use the Department's rotary-wing UAV as the subject of the research that was conducted. Professor Dobrokhodov's knowledge, experience, and interest in this research were critical to overcoming many of the challenges that were encountered along the way. Professor Dobrokhodov's contributions to this thesis, and his dedication to my learning experience, is the foundation for the successful completion of this research. Dr. Ioannis Kitsios made himself available and provided the guidance and support, without which, this research endeavor would not have been possible. I also want to thank Professor Kevin Jones, of the MAE Department, for providing insight and expertise on some of the mechanics of the miniature UAV. I am also indebted to the support and software engineers at MathWorks who made the final version of the Simulink model possible by providing tireless and reliable software support.

Lastly, I want to thank the one who deserves the full credit for where I am today, my heavenly Father. Without Him, I would not be here today nor would I have the loving and supportive family that I love. To my wife, Carol, thank you for your tireless support and dedication to our success. To my daughters Angelica and Sara, life has reason and purpose because of you. To our new baby boy, due February 2010, I cannot wait to hold you in my arms.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. MOTIVATION

Rotary-wing Unmanned Aerial Vehicles (RW UAV) provide capabilities that fixed-wing UAVs cannot. Unlike fixed-wing UAVs, RW UAVs provide enhanced flexibility in the joint Navy, Marine Corps, Army, and Air Force battle space by enabling flexibility in aerial reconnaissance. In contrast to the fixed-wing UAVs, RW UAVs provide vertical take-off and landing (VTOL), thereby foregoing the necessity for cleared terrain. Additionally, RW UAVs provide the ability to conduct personnel clearing in urban terrain to a degree that is not possible with fixed-wing UAVs.

The advantages provided by a RW UAV make it highly desirable for DoD applications. Development of a nonlinear RW UAV software model will facilitate future development of RW UAV control design at NPS.

B. RESEARCH OBJECTIVES

The overall objective of the research conducted for this thesis is to provide a simulation environment that will facilitate and enable future work such as Hardware in the Loop (HIL) testing of the Micro Pilot autopilot (MP AP) and flight testing of the MP AP on the physical plant. Micro Pilot software exists for testing of controller code; however, this software is not specific to the MAE Department's UAV model (T-Rex Align 600), nor is it configurable to any other miniature RW platform. The simulation environment will simulate the 6-Degree of Freedom (6-DOF) model of the MAE Department's RW UAV and will provide a test environment that is configurable to any specified model.

This research effort focuses on the implementation of existing helicopter dynamics theory to develop a non-linear 6-DOF RW UAV model using the MathWorks Simulink modeling environment. Emphasis is placed on obtaining results that are in agreement with the available literature covering helicopter dynamics theory.

C. T-REX ALIGN 600 MODEL DESCRIPTION

The T-Rex Align 600 is a remote controlled model helicopter. The main rotor is composed of two rotor blades, made of carbon fiber, and a stabilizer bar for active rate damping. The T-Rex Align uses an electric power system and a lightweight carbon fiber frame. The canopy is made of fiberglass and the tail rotors, like the main rotors, are made of carbon fiber material. Figure 1 illustrates the side view of the T-Rex Align 600 air frame.

Figure 1. T-Rex Align 600 Model (Side View)

The T-Rex Align 600 plant properties are outlined in Table 1.

PROPERTY	QUANTITY
Flying Mass/ Weight	3 kg (29.4 N)
Main Rotor Radius	.6 m
Main Rotor Diameter	1.35 m
Tail Rotor Diameter	.24 m

Table 1. T-Rex Align 600 Plant Properties

D. APPROACH

Helicopter dynamics is dominated by the forces and moments generated by the helicopter main rotor and tail rotor. First, pilot input to rotor blade responses are analyzed and modeled. Second, rotor aerodynamics is considered in order to develop predictable rotor thrust responses. Also, main Rotor flapping dynamics are analyzed and modeled. The main rotor flapping is what generates translational acceleration and angular rotations of the helicopter body. Once the main rotor thrust and flapping is modeled, all forces and moments are summed and applied at the helicopter body center of gravity, and vehicle motion is simulated using MathWorks Simulink simulation environment. Flight Gear software is incorporated with the Simulink model to provide motion animation.

There is extensive documentation of previous modeling efforts that have been conducted. The approach of this thesis effort relies on the modeling of known parameters for the MIT instrumented X-Cell SE helicopter used in [1] and [2]. The intent is to model a helicopter similar to the ME Department's T-Rex Align in order to provide a platform from which future work in system identification can uncover the specific model parameters for this particular UAV. Model parameters for the MIT X-Cell SE are listed in Appendix A.

After completion of the modeling, the nonlinear model is linearized about the hovering flight condition in order to uncover the state boundaries about which the controller will be useful. After linearization, the PD gains are estimated, tuned, and tested for flight-path tracking performance. The flight paths that the model will be expected to fly are the square and figure-8 flight paths. Finally, the flight path tracking performance results are discussed.

THIS PAGE INTENTIONALLY LEFT BLANK

II. MODELING INTRODUCTION

A. BODY-FIXED FRAME AND HELICOPTER VARIABLES

The modeling of the RW UAV uses only one body centered reference frame. Figure 2 illustrates the helicopter body-fixed reference frame with the origin at the center of gravity. The variables are shown on the body axes x, y, z. The x-axis is positive pointing away from the nose of the UAV, the y-axis is positive towards the starboard side, and the z-axis is positive downward. The variables include the body velocities u, v, w, the Euler angles ϕ, θ, ψ, and the body rates p, q, r.

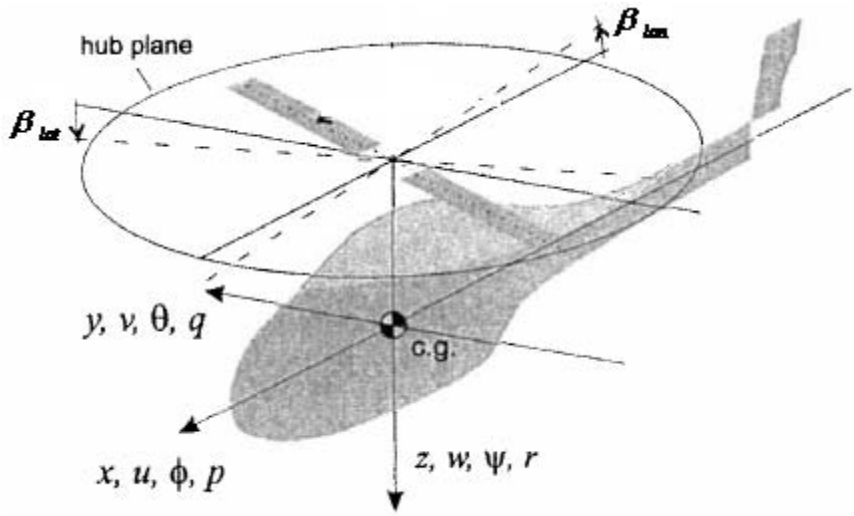


Figure 2. Helicopter Body-Reference Frame (After [2])

Positive rotations are defined using the right-hand rule, with the thumb pointing in the positive axis direction. The rotation is defined by curling the other four fingers in the direction of rotation. For example, with the right thumb pointing in the positive z direction (pointing down), a positive yaw angle is defined by a clockwise body rotation.

The main rotor disk can tilt about the rotor hub along two degrees of freedom, about the lateral (y-axis) and longitudinal (x-axis) directions. The flap angles β_{lat} and β_{lon} represent the main rotor flap angles about the lateral and longitudinal directions,

respectively. A positive lateral flap angle (β_{lat}) occurs when the main rotor tilts towards the positive y-axis because this corresponds to a rotation in the direction of positive roll angle (ϕ). A negative longitudinal flap angle (β_{lon}) occurs when the main rotor tilts towards the positive x-axis because this corresponds to a negative pitch angle (θ). Note that a positive pitch angle (θ) is defined by placing the right hand face up, with the right hand thumb pointing towards the right (+y direction). Curling the fingers in towards the hands defines the positive pitch rotation.

B. RIGID-BODY EQUATIONS OF MOTION

The helicopter is able to simultaneously translate and rotate about all 6-DOF. The equations of motion for the fuselage six degrees of freedom are derived by Gavrilets in [1]. Summing the moments and forces, due to the main rotor, tail rotor, and aerodynamic forces on the fuselage, gives the three translational equations of motion as functions of the UAV velocity states (u, v, w) and the forces acting on the body center of gravity (X_{MR}, X_{fus}, X_{tr}, for example) [1, p. 33]:

$$\dot{u} = vr - wq + \frac{(X_{MR} + X_{fus} + X_{tr})}{m} - g\sin\theta \tag{2.1}$$

$$\dot{v} = wp - ur + \frac{(Y_{MR} + Y_{fus} + Y_{tr})}{m} + g\cos\theta\sin\phi \tag{2.2}$$

$$\dot{w} = uq - vp + \frac{(Z_{MR} + Z_{fus} + Z_{tr})}{m} + g\cos\theta\cos\phi \tag{2.3}$$

The X term, for example, denotes a force in the x direction, while the subscripts MR, fus and tr represent the forces in the x direction due to the main rotor, fuselage and tail rotor, respectively. The term g is the gravity force.

Assuming the cross products of inertia are small gives the three equations of motion describing the body-angular motion as a function of the body rotation rates ($p,\ q,\ r$) and moments, $L_{MR},\ L_{fus},\ L_{tr}$ acting on the body center of gravity:

$$\dot{p} = \frac{qr(I_{yy} - I_{zz})}{I_{xx}} + \frac{\left(L_{MR} + L_{fus} + L_{tr}\right)}{I_{xx}} \tag{2.4}$$

$$\dot{q} = \frac{pr(I_{zz} - I_{xx})}{I_{yy}} + \frac{\left(M_{MR} + M_{fus} + M_{tr}\right)}{I_{yy}} \tag{2.5}$$

$$\dot{r} = \frac{pq(I_{xx} - I_{yy})}{I_{zz}} + \frac{\left(N_{MR} + N_{fus} + N_{tr}\right)}{I_{zz}} \tag{2.6}$$

All of the translational accelerations, velocities, angular rates, and Euler angles will be generated using the Simulink 6-DOF block with the main rotor, tail rotor, drag, and aerodynamic forces and moments as inputs. Figure 3 illustrates the forces and moments acting on the UAV body. Note that the main rotor thrust T=$\begin{bmatrix} X_{MR} & Y_{MR} & Z_{MR} \end{bmatrix}^T$ and the drag force on the body is $F_{fus} = \begin{bmatrix} X_{fus} & Y_{fus} & Z_{fus} \end{bmatrix}^T$. The tail rotor thrust $\left(T_{tr}\right)$ is modeled to act normal to the fin plane.

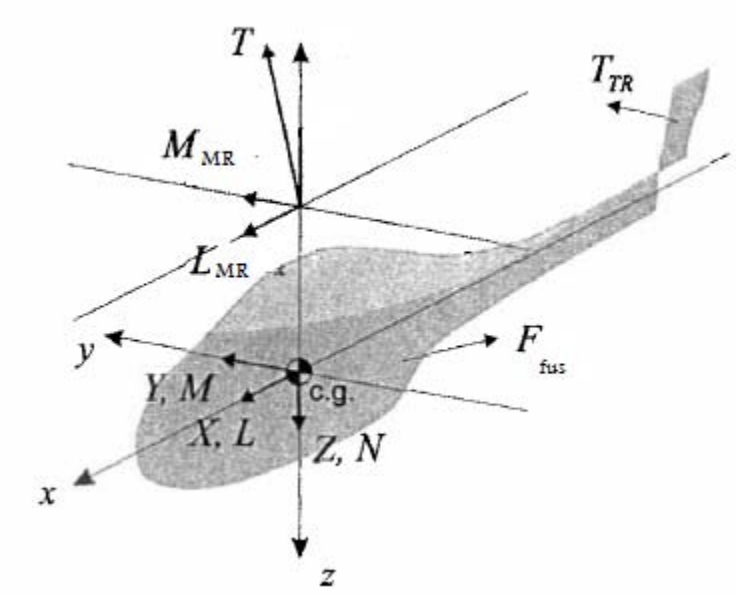


Figure 3. Helicopter Forces and Moments (After [2])

The outputs to the 6-DOF block will be the center of gravity's position, velocity, Euler angles, and Euler rates. These are derived by the 6-DOF block by implementing the state dynamic equations (2.1) through (2.6).

In summary, the model has 14 vehicle states:

$$\underline{x} = \begin{bmatrix} x & y & z & \phi & \theta & \psi & u & v & w & p & q & r & \beta_{lon} & \beta_{lat} \end{bmatrix}$$

with a body-fixed coordinate system at the center of gravity. Positive rotations are defined using the right hand rule and the forces and moments are summed at the vehicle center of gravity. The Simulink 6-DOF block is used to generate the vehicle state dynamics.

The following chapter will discuss the modeling of the necessary components that will generate the forces and moments on the vehicle center of gravity, as well as the modeling of the channel for pilot input to control surface command.

III. ROTARY-WING UNMANNED AERIAL VEHICLE (RW UAV) MODELING

A. CONTROL INPUT TO CONTROL SURFACE COMMAND DESIGN

There are four inputs that a pilot/operator can give to the UAV. The first input is collective and, in full-scale helicopters, is provided through a lever. This lever is an elongated version of what looks like the emergency brake in a car. A change in the lever position will provide a change in the main rotor collective, or the main rotor blade angle of attack. This is accomplished through a variable orientation swash plate. As the pilot provides collective input, the swash plate changes accordingly. A collective lever input causes the swash plate to rise up or down while maintaining a flat orientation. In the UAV controller, collective is controlled through one of the channel stick inputs, normally set at full down by an internal spring force.

The second type of input available to the pilot/operator is the cyclic. In a full-scale helicopter, this is provided by the stick immediately to the front of the pilot, which can rotate by varying degrees. The cyclic input can be classified by longitudinal cyclic and lateral cyclic. Longitudinal cyclic is a cyclic stick input that affects the translational acceleration of the vehicle along the x-axis. Longitudinal cyclic commands also affect the pitch orientation of a helicopter. This is achieved by providing a forward or aft stick command. A full forward stick command causes the main rotor blade angle to be a maximum when it reaches the $-x$-axis and cycles through to a minimum angle of attack when the blade reaches the $+x$-axis. In a full-scale helicopter, this is achieved by a tilting of the swash plate, which in turn, causes cyclic (variable) blade angle commands as the rotor blade rotates around its circular trajectory.

Lateral cyclic is a cyclic stick input that affects the translational acceleration of the vehicle along the y-axis. Lateral cyclic also affects the helicopter roll orientation. This is achieved by providing left and right stick commands. A full right stick command causes the main rotor blade angle to be a maximum when the rotor blade reaches the $-y$-axis; as the blade cycles to the $+y$-axis, along its circular trajectory, the blade angle

becomes increasingly smaller until it reaches a minimum at the $+y$-axis. The maximum blade angle at the –y position is what generates a higher lift force on the blade at this position, when compared to the lift of the blade at the $+y$ position. This difference in lift is what causes the main rotor disk plane to flap. As the rotor blade achieves its maximum angle of attack at the –y position, the main rotor tilts in the +y direction, and this causes the helicopter to accelerate in the +y direction. Again, this is physically achieved through the mechanical tilting of the swash plate, which in turn controls the cyclic behavior of the rotor angle of attack.

The last form of pilot command is the rudder pedal. The rudder pedal provides collective input commands to the tail rotor, much like the collective lever provides collective commands to the main rotor. The thrust generated by the tail rotor is necessary for canceling the torque on the inertial frame that is generated by the main rotor.

Table 2 outlines the four different command types and their physical meaning on a handheld R/C controller.

Command **(Simulink Pilot Joystick Blockset ID)**	**Stick Input Value Range**	**Physical Meaning on Handheld Controller**
Collective Input (Throttle)	[0,1]	[Full throttle down, FT up]
Longitudinal Cyclic (Pitch)	[-1,1]	[Full Back Stick, Full Fwd Stick]
Lateral Cyclic (Roll)	[-1,1]	[Full Left Stick, Full Right Stick]
Rudder Collective (Yaw)	[-1,1]	[Full Left Pedal, Full Right Pedal]

Table 2. Joystick Block Value Range Outputs

The full positions provide maximum acceleration that corresponds to the specified input. Prouty [3] outlines several helicopters and their collective, cyclic, and tail rotor angle of attack ranges. These are listed in Table 3 and provide references for choosing initial estimates for the model collective and cyclic ranges.

Aircraft	MR Collective	TR Collective	Lateral Cyclic	Longitudinal Cyclic
Agusta A109	0° – 16°	–7° / 21°	±6.25°	–10.5° / 12.5°
Bell 206L-3	6° – 22°	–12.45° / 19.5°	±10°	±10°
Bell 412	0° – 16°	–10.15° / 19.85°	–12.6° / 6.7°	±11°
MBB BO105 LS	–.2° / 15°	–8° / 20°	–5.7° / 4.2°	–11° / 6°
Robinson R22 Beta	1.5° – 14.5°	–10.6° / 19.5°	–9.5° / 6°	–9° / 11°

Table 3. Sample of Full-Scale Aircraft Control Ranges (After [3])

The response ranges that are used for the T-Rex Align model are listed below:

X-Cell 60 Simulink Model Rotor Response Ranges

MR Collective	TR Collective	Lat Cyclic	Long Cyclic
–5° / +15°	–20° / +25° ˚	±20°	±20°

The Simulink model is designed so that no input is required to maintain the vehicle in hovering trim. Also, the cyclic limits are higher than the average full scale limit since the miniature UAV is more maneuverable and aerobatic.

The first aspect of the model to be designed is the transformation that takes place from the stick control input [-1,1], for example, to the control surface command. The stick input with signal -1 corresponds to a full left stick input while a control input signal of 1 corresponds to a full right stick input; a control signal of 0 corresponds to the neutral position.

The stick position [-1,1] is coded in the delay time between pulses of the remote controller's baseband signal. The radio signal illustrated in Figure 4 corresponds to the delay time for each of five stick positions. The long delay at the end of the signal is a reset for the next set of pulses. The receiver onboard the UAV splits the information from each input and sends it to the corresponding servo, in turn. Each of the servos receive as inputs a pulse width modulated signal, with duty cycle information, typically in the range from $600\,\mu s$ to $2400\ \mu s$. The servo then decodes the duty cycle information and generates a mechanical rotation from $-45°$ to $+45°$. The servo's mechanical output is proportional to the duty cycle. A duty cycle input to the servo of $600\ \mu s$, for example,

will give a servo output of $-45°$; a duty cycle input of 1,520 μs will give an output of $0°$, and a duty cycle input of 2,400 μs will give an output of $+45°$. All angles in between can be achieved by varying the PWM duty cycle inputs.

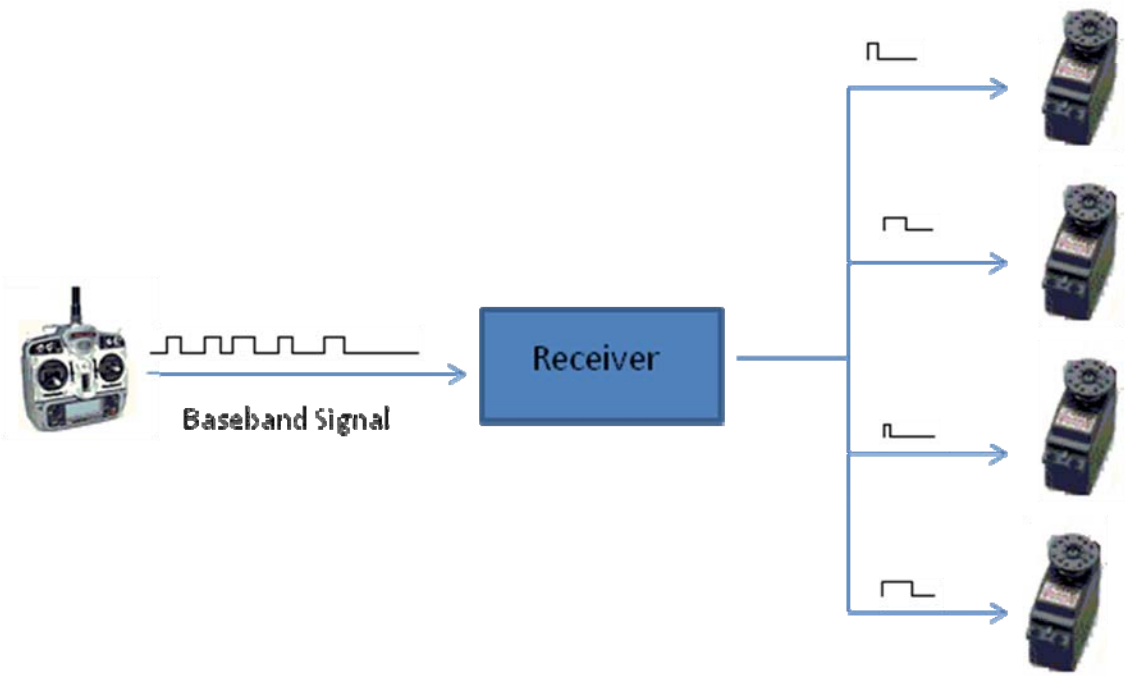


Figure 4. Signal Flow (Transmitter to Servo Output)

The servo rotations are then transmitted through linkages that control the helicopter control surfaces. More specifically, the main rotor linkages come from the cyclic (lateral and longitudinal) servos and from the collective servo. Figure 5 illustrates the signal flow from the servo input to the commanded flap angle as well as the signal flow from the commanded flap angle to the actual flap angle. The linear flapping dynamics are explained in 3.B.7.

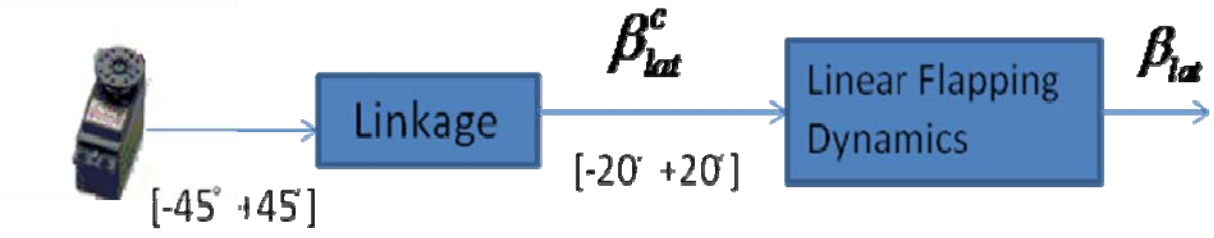


Figure 5. Signal Flow (Servo to Commanded Flap Angle)

The linkage that transforms the servo mechanical rotation to a control surface command is linear and proportional to the servo mechanical rotation.

Figure 6 illustrates the model, which simulates the channel, from control input $u = \begin{bmatrix} u_{lat} & u_{lon} & u_{col} & u_{ped} \end{bmatrix}$ to PWM duty cycle with limits $D_c = \begin{bmatrix} 600 & 1520 & 2400 \end{bmatrix} \mu s$.

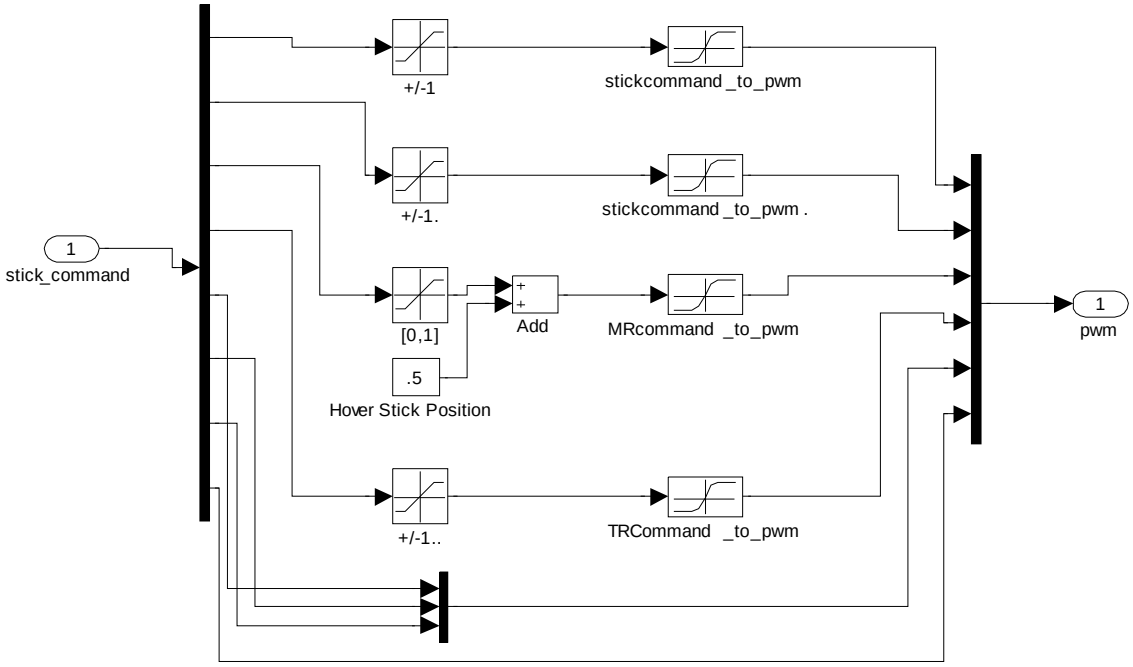


Figure 6. Channel Model.

The first four signals in Figure 6 represent the command inputs u; the last several signals are other signals that are being forwarded to the model. Saturation limits are placed according to the control input limits. The .5 constant on the third signal ensures that when the control effort is zero, the UAV will maintain a 50% collective input for hovering flight. This is because the main rotor collective is designed for hovering at 50% command input. Figure 7 illustrates the model of the helicopter servos with PWM duty cycle as inputs and mechanical servo rotations as outputs.

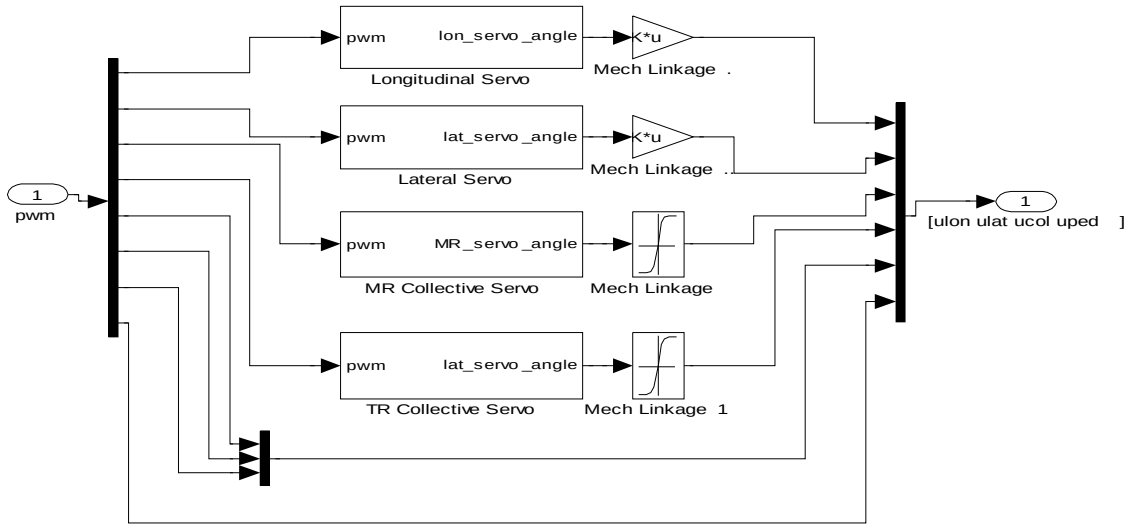


Figure 7. Servo Model (Control Input to PWM)

Each servo block also has second order filters that model the time responses of the mechanical output. The time response models for the servos were obtained from [1, p. 52]. The time response model for the servo mechanical output is illustrated in Figure 8.

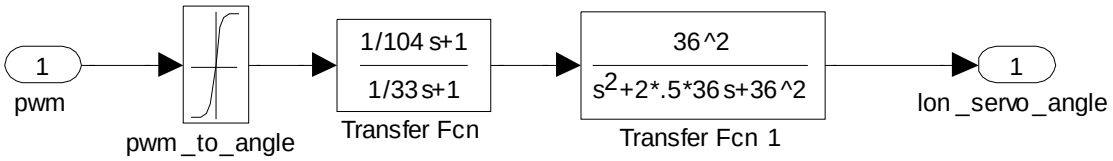


Figure 8. Servo Input to Output Filter

Figure 9 illustrates the PWM, servo mechanical rotation, and control surface command outputs from several longitudinal step commands of -1, 0, and 1. These inputs cover the full range of stick input motion for longitudinal control.

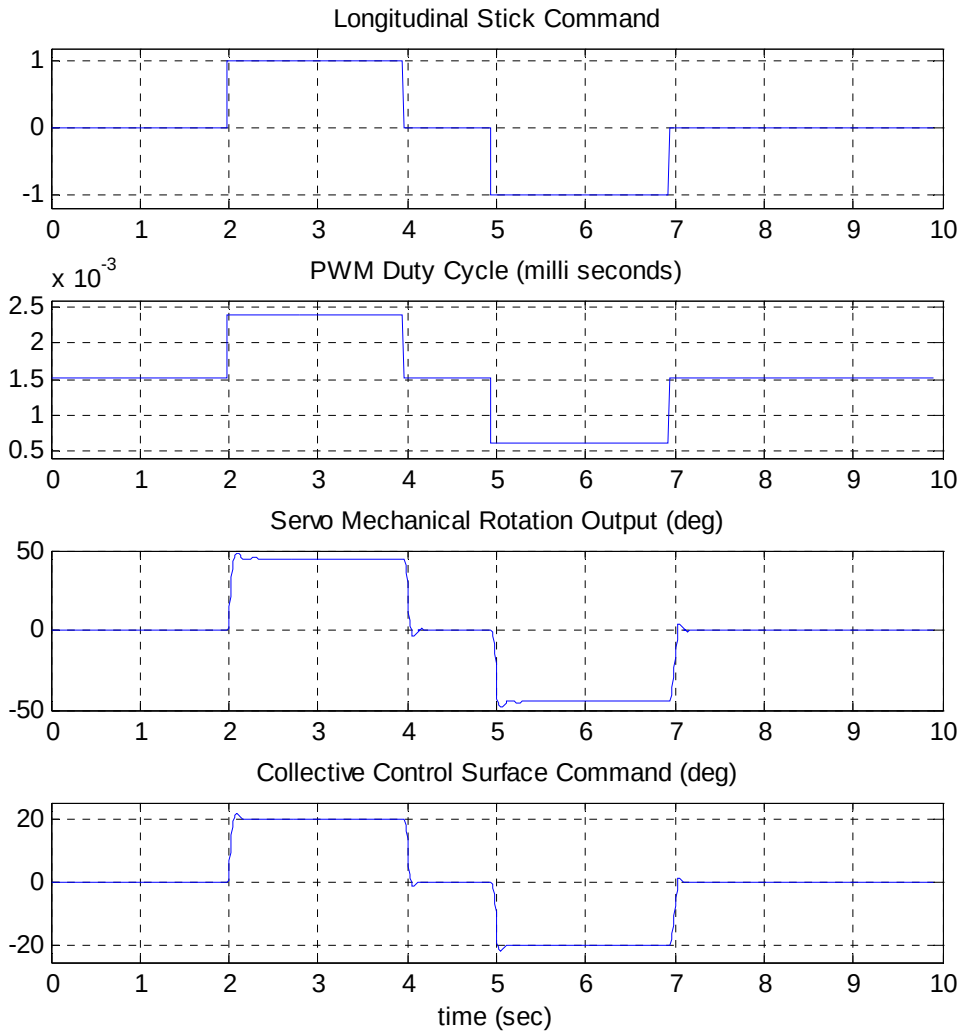


Figure 9. Channel Model Responses

Initially, when the input is zero, the duty cycle output is in the middle of the duty cycle range, 1520 μs, which gives the zero position on the servo and no control surface command. When the longitudinal input is maxed out at full forward (+1), the duty cycle maxes out at 2400 μs and this generates the maximum servo rotation of $+45°$ and a proportional cyclic control surface command of $20°$. It is evident that the full back command (-1) generates the $-20°$ collective control surface command. The second order filter models the second order behavior of the servo mechanism and gives a settling time of .055 seconds with 7.6% overshoot.

The duty cycle limits, servo mechanical outputs, and other parameters that are accessed by the table blocksets in the models (illustrated in Figures 5 through 7) are defined in "Config.m" outlined in Appendix B. This file also defines other constant parameters like the UAV mass, dimensions, moments of inertia, and so on.

B. MAIN ROTOR DESIGN

The main rotor thrust is directly related to the pilot commanded collective and indirectly related to the vehicle velocity states (u, v, w) and the rotor inflow velocity (v_i). This relation is given by Padfield [4] and Leishman [5] as:

$$T = \left[\theta_o \left(\frac{1}{3} + \frac{\mu^2}{2} \right) - \left(\frac{\mu_z + \lambda_i}{2} \right) \right] \frac{as}{2} \rho \left(\Omega R \right)^2 A_d \tag{3.1}$$

Determination of the thrust is complicated by the fact that the rotor inflow, λ_i, is directly influenced by the thrust. Padfield explains that as the rotor thrust increases, the rotor inflow also increases [4, p. 122]. The opposite is also true. That is, as the rotor thrust decreases, the main rotor inflow velocity decreases. Knowing this helps the designer to understand how the thrust and inflow interact and the ultimate behavior of the thrust to a pilot commanded step in collective, which is denoted by θ_o.

If, for example, the pilot gives a step collective command, increasing θ_o, the thrust will begin to increase. This occurs because the rotor inflow term, λ_i, has not had enough time to increase to the critical value that will cause the thrust to decrease to a new equilibrium. So, during this phase of the thrust response, the thrust is increasing since the change in θ_o is larger than any increase in λ_i; note that the λ_i term is being subtracted in Equation (3.1). Later, it will be shown how the rotor inflow increases with increasing thrust but, for now, the thrust response to an increasing inflow will be analyzed. As the rotor inflow velocity increases, the λ_i term gets larger. The same is true for the μ_z term when the vehicle is climbing. This causes the thrust to begin to decrease once λ_i reaches

a critical value. The thrust will continue to decrease until it reaches a new equilibrium point. To better understand this behavior, the process for determining the rotor inflow will now be discussed.

1. Rotor Inflow from Momentum Theory

Momentum analysis in forward flight provides for the solution of the main rotor inflow in forward flight as well as in climb. Due to the complexity of the main rotor inflow in descent, a different technique will be used for this flight configuration.

The relationship between rotor thrust and the rotor inflow is dictated by the conservation of mass and the conservation of momentum and energy [4, pp. 115–116]. The theory that allows for the derivation of these relationships is known as momentum theory. Conservation of mass dictates that the mass flow rate of a volume of air passing through the main rotor disk area is proportional to the volume flow rate (m^3/sec) and the air density. This relation is given by Anderson as [6, p. 93]:

$$\dot{m} = \rho(V_n A_d) \tag{3.2}$$

where V_n is the normal component of the velocity of the mass of air through the rotor disk. The thrust generated from the acceleration of the air through the main rotor is simply

$$T = ma \tag{3.3}$$

Padfield [4] relates the rate of change of momentum between the undisturbed upstream conditions and the far wake to the rotor loading and is given by [4, p. 116]:

$$T = \dot{m} v_{i\infty} \tag{3.4}$$

where $v_{i\infty}$ is the induced flow far downstream of the rotor. Padfield proceeds to relate the change in kinetic energy of the flow to the work done by the rotor, leading to the conclusion that the velocity of the air far downstream of the rotor is twice the induced velocity of the air immediately upstream of the rotor blades. This gives the expression [4, p. 116]:

$$v_{i\infty} = 2v_i \tag{3.5}$$

Substituting Equations (3.2) and (3.5) into (3.4) gives [4, p. 116]:

$$T = (\rho A_d V_n) \times (2v_i) \tag{3.6}$$

The velocity component normal to the rotor depends on the helicopter flight configuration. In the hover flight configuration, the upstream velocity of the air normal to the rotor is the same as the inflow velocity, therefore, $V_n = v_i$ in a hovering flight configuration. In a climbing flight configuration, the upstream velocity of the air normal to the rotor is $V_c + v_i$. In a descent, the upstream velocity of the air normal to the rotor is $V_d - v_i$ [4, p. 116]. At low forward airspeeds, the normal component is given by Prouty as $\sqrt{V^2 + v_i^2}$ [4, p. 123]. Higher airspeed forward flight is the more general case and simpler to use because the normal component velocity reduces to $\sqrt{V^2 + v_i^2}$ at the lower airspeeds. The velocity component normal to the disk rotor for all forward airspeeds is given by Leishman as [5, p. 63]:

$$\begin{aligned} &(\textit{Hovering and Forward Airspeed}) \\ &V_n = \sqrt{V^2 + 2Vv_i \sin\alpha + v_i^2} \end{aligned} \tag{3.7}$$

Where α is the angle between the horizon and the main rotor blade tip path plane. Equation (3.7) is useful because it reduces to the normal velocity component for hovering flight as well as forward flight when the helicopter is in either of these configurations. Rearranging (3.6) for v_i gives:

$$v_i = \frac{T}{2\rho A_d V_n} \tag{3.8}$$

When hovering, the velocity of the air normal to the rotor (V_n) is equal to the rotor inflow (v_i). Substituting v_i for V_n in (3.8) gives $T = v^2_{ihover} 2\rho A_d$. Substituting this into (3.8) gives:

$$v_i = \frac{T}{2\rho A_d V_n} = \frac{v_{ihover}^2 \left(2\rho A_d\right)}{\left(2\rho A_d\right) V_n} = \frac{v_{ihover}^2}{V_n} \tag{3.9}$$

Substituting (3.7) into (3.9) gives the main rotor inflow for any airspeed and orientation as [5, p.64]:

$$v_i = \frac{v^2{}_{ihover}}{\sqrt{\left(V_\infty \cos\left(\alpha\right)\right)^2 + \left(V_\infty \sin\left(\alpha\right) + v_i\right)^2}} \tag{3.10}$$

The velocity parallel to the main rotor plane is $V_\infty \cos\left(\alpha\right)$. Normalized by the main rotor tip speed $\left(\Omega R\right)$ gives the normalized translational velocity $\mu = V_\infty \cos\left(\alpha\right) / \left(\Omega R\right)$. The main rotor inflow ratio is [5, p. 65]:

$$\lambda = \frac{V_\infty \sin\left(\alpha\right)}{\Omega R} + \frac{v_i}{\Omega R} \tag{3.11}$$

Leishman then writes the first term in (3.11) in terms of the normalized normal airspeed μ by relating $V_\infty \sin\left(\alpha\right) = V_\infty \cos\left(\alpha\right) \tan\left(\alpha\right) = \mu \tan\left(\alpha\right)$. Therefore, Equation (3.11) is now

$$\lambda = \mu \tan\left(\alpha\right) + \frac{v_i}{\Omega R} \tag{3.12}$$

Substituting (3.10) into (3.11) gives

$$\lambda = \mu \tan\left(\alpha\right) + \frac{\lambda_{hover}^2}{\sqrt{\mu^2 + \lambda^2}} \tag{3.13}$$

The inflow ratio at hover is given by [4]:

$$\lambda_{hover} = \sqrt{\frac{C_T}{2}}$$

The solution to the inflow ratio is then [4, p. 65]

$$\lambda = \mu \tan(\alpha) + \frac{C_T}{2\sqrt{\mu^2 + \lambda^2}} \tag{3.14}$$

Leishman explains that a numerical solution to Equation (3.14) can be implemented using the Newton-Raphson technique. The advantage to using this method is a more rapid convergence as well as the fact that it can be employed for practical calculations involving rotors in all ranges of flight configurations, from hover to climb, as well as forward flight and descents. Equation (3.14) does provide convergence in descents; however, the solution may not be physically realizable. In descent cases, the rotor inflow velocity will be approximated as linear and will be discussed in Chapter III.B.4.

2. Newton-Raphson Iteration Technique

The Newton-Raphson iteration scheme is implemented in the estimation of the main rotor and tail rotor inflow. The inflow is iteratively estimated from the previous value until the error between the current and previous value is less than some acceptable tolerance. The inflow is determined iteratively using Equations (3.15) through (3.18) [5, p. 67]:

$$\lambda_{n+1} = \lambda_n - \left[\frac{f(\lambda)}{f'(\lambda)} \right]_n \tag{3.15}$$

$$f(\lambda) = \lambda - \mu \tan(\alpha) - \frac{C_T}{2\sqrt{\mu^2 + \lambda^2}} \tag{3.16}$$

$$f'(\lambda) = 1 + \frac{C_T}{2}\left(\mu^2 + \lambda^2\right)^{-3/2} \lambda \tag{3.17}$$

The iteration scheme continues until the error estimator is less than .005%. The error estimator is given as [5, p. 66]:

$$\varepsilon = \left| \frac{\lambda_{n+1} - \lambda_n}{\lambda_{n+1}} \right| \tag{3.18}$$

For any given thrust, Equation (3.15) will converge to the correct value of inflow by using an initial inflow estimate of $\lambda_o = \lambda_{hover}$. The hover inflow λ_{hover} is determined at hovering conditions, where the main rotor thrust is known to be equal to the weight of the helicopter. Since $V_n = v_i$ at hover, Equation (3.8) can be rewritten as:

$$v_i = \frac{T}{2\rho A_d V_n} \quad \Rightarrow \quad v_i^2 = \frac{T}{2\rho A_d} \quad \Rightarrow \quad v_{ihover} = \sqrt{\frac{T}{2\rho A_d}} \tag{3.19}$$

where the weight of the helicopter is $W = 8.2 \times 9.806 = 80.41$ N . The main rotor disk area is $A_d = \pi R^2 = \pi \times (.775)^2 = 1.887$ m^2, which gives a main rotor hovering inflow of $v_i = 4.171$ m/s . Normalized by the rotor tip speed (ΩR) gives $\lambda_{hover} = 32.23 \times 10^{-3}$.

The following example shows the inflow convergence for both cases where the thrust is increasing and decreasing. Initially, the main rotor thrust increases with an increase in collective lever input. Later, it will be shown how this occurs, but for now it will be assumed that the main rotor thrust for two consecutive samples are T=[80.5 81.3]. There is no translational velocity component, so $\mu = 0$ and the tip path plane angle, $\alpha = 0$. The coefficients of thrust for these two samples are obtained by normalizing the thrust values of each sample by $\rho(\Omega R)^2 \pi R^2$ and gives $C_T = \begin{bmatrix} 2.079 \times 10^{-3} & 2.0997 \times 10^{-3} \end{bmatrix}$. Running these thrust values through "getvi.m" outlined in Appendix B, gives the two convergence trajectories illustrated in Figures 9 and 10. The m files used for generating convergent inflow velocities for the main rotor and tail rotor are listed in Appendix B.

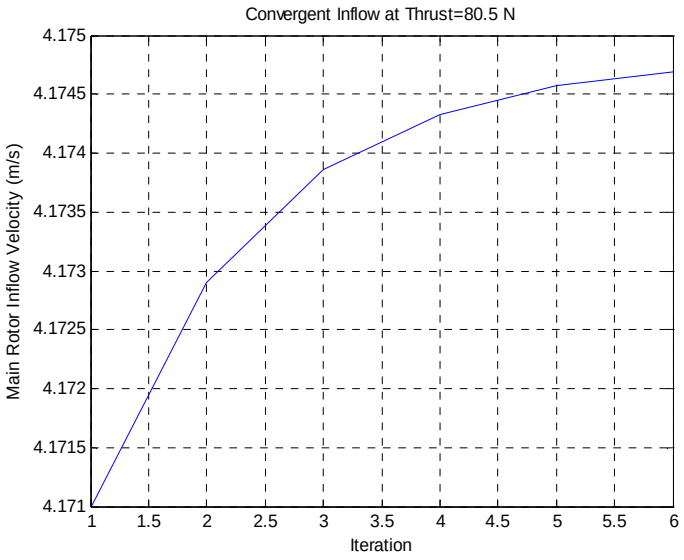


Figure 10. Convergent Inflow Using Newton-Raphson Technique (T=80.5 N)

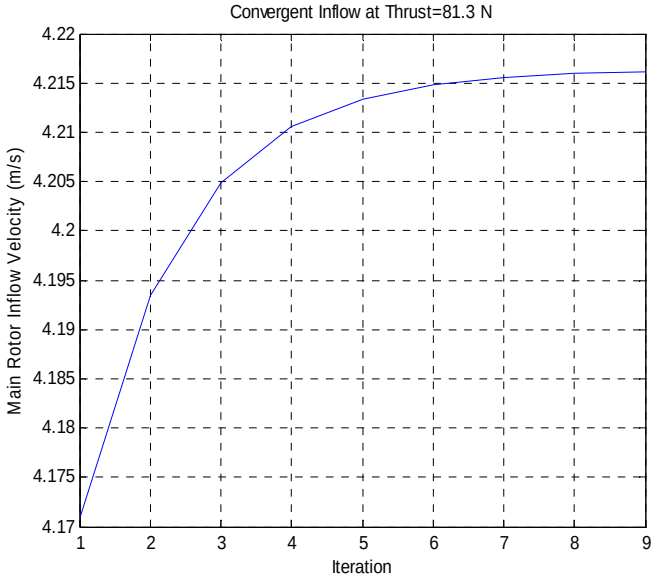


Figure 11. Convergent Inflow Using Newton-Raphson Technique (T=81.3 N)

Figure 10 illustrates the value of the estimated rotor inflow for up to 6 iterations. The inflow estimate starts at $v_i = 4.171$ m/s, before the first iteration. The Newton-Raphson iteration loop in "getvi.m" increases the value of v_i, using Equations (3.15) through (3.17), until $\varepsilon < .005\%$. As soon as the error becomes less than .005%, the value

of the inflow for that iteration routine is extracted and used to compute the new value of thrust. From this example, it is clear that as the thrust increases the main rotor inflow velocity also increases. The main rotor inflow increases from the initial estimate of 4.171 m/s to 4.175 m/s, which is the new rotor inflow at T=80.5 N. As the thrust increases to 81.3 N on the second sample, the rotor inflow converges to 4.215 m/s, as illustrated in Figure 11. Figure 12 illustrates the main rotor inflow convergence when the rotor thrust decreases.

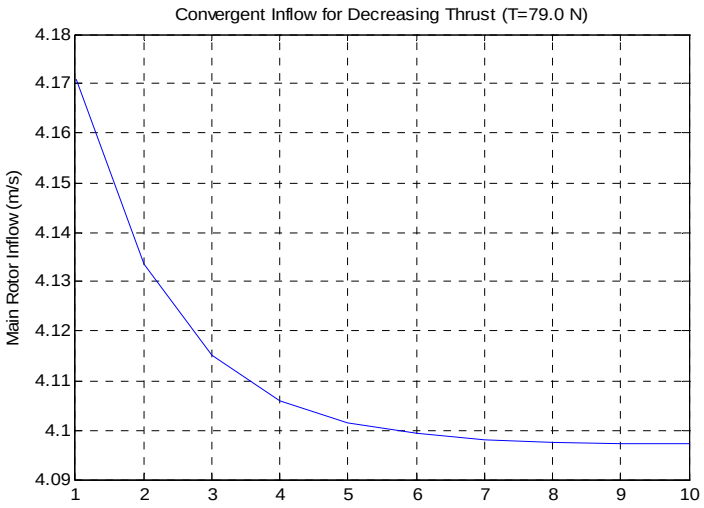


Figure 12. Convergent Inflow Using Newton-Raphson Technique (T=79.0 N)

In this case, as the thrust decreases to 79.0 N, the main rotor inflow velocity decreases to 4.0971 m/s. This iteration scheme is implemented in the Simulink model for both the main rotor and tail rotor at a rate of 100 Hz.

3. Main Rotor Thrust and Inflow Response

The helicopter simulation starts in a hovering configuration since the inflow and thrust values are known in this regime. As the main rotor collective is changed, the rotor inflow is determined as described in section III.B.2. The convergent inflow is then used in Equation (3.1) to calculate the main rotor thrust.

As the pilot/operator gives a collective input command (θ_o), the new value of thrust is determined from Equation (3.1) by using the previous known value of the inflow (1 sample delay). Then, the new value of thrust is input into the Newton-Raphson technique to generate the new value of the inflow and the process starts over again; the loop is closed in this fashion.

As explained earlier, Equation (3.1) reveals that as the inflow increases in response to an increase in thrust, the thrust will begin to decrease because of the $\left(\frac{\mu_z + \lambda_i}{2}\right)$ term that is being subtracted. A simulation was run in order to illustrate this behavior. The input is a step collective from $5.495°$ to $5.657°$. The response of the thrust and main rotor inflow velocity is illustrated in Figure 13.

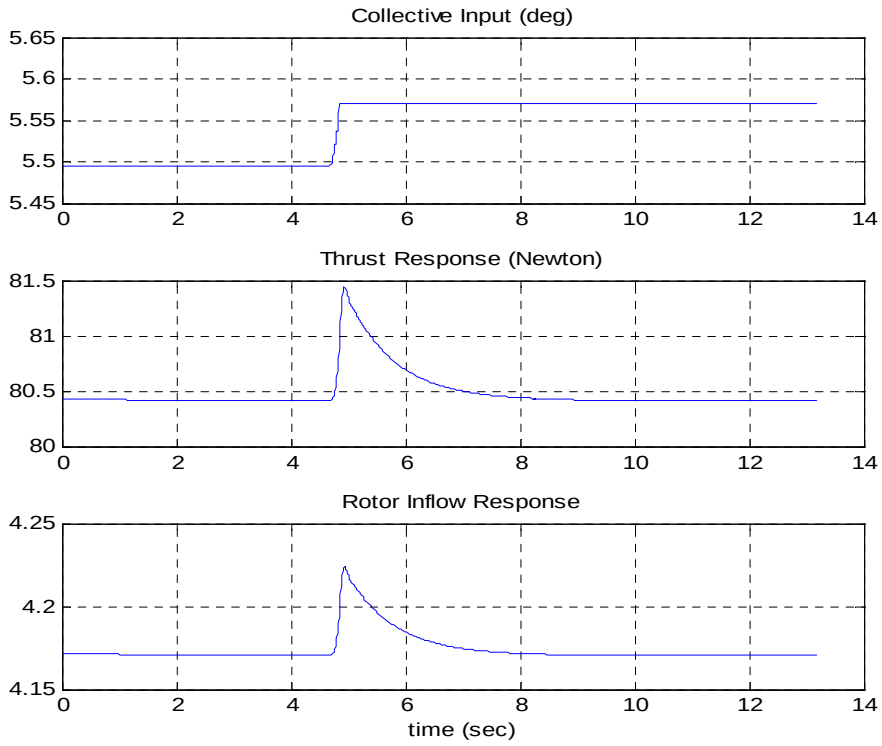


Figure 13. Main Rotor Thrust and Inflow Response to Step Collective

The thrust response illustrated in Figure 13 shows that the thrust returns to the equilibrium thrust. This response is in agreement with [4, p. 305], which illustrates the thrust response to a step collective for a full-scale helicopter. In this case, the equilibrium

value of the thrust is the weight of the vehicle. This behavior guarantees the first order heave response that is characteristic of both full-scale and miniature helicopters. That is, for any given step collective input, the velocity rate in the z direction will display a second-order, over damped response. Figure 14 illustrates the heave rate response for a collective step input from $5.495°$ to $6.5°$. This heave response is also in agreement with the vertical axis response characteristic detailed in [4, p. 401].

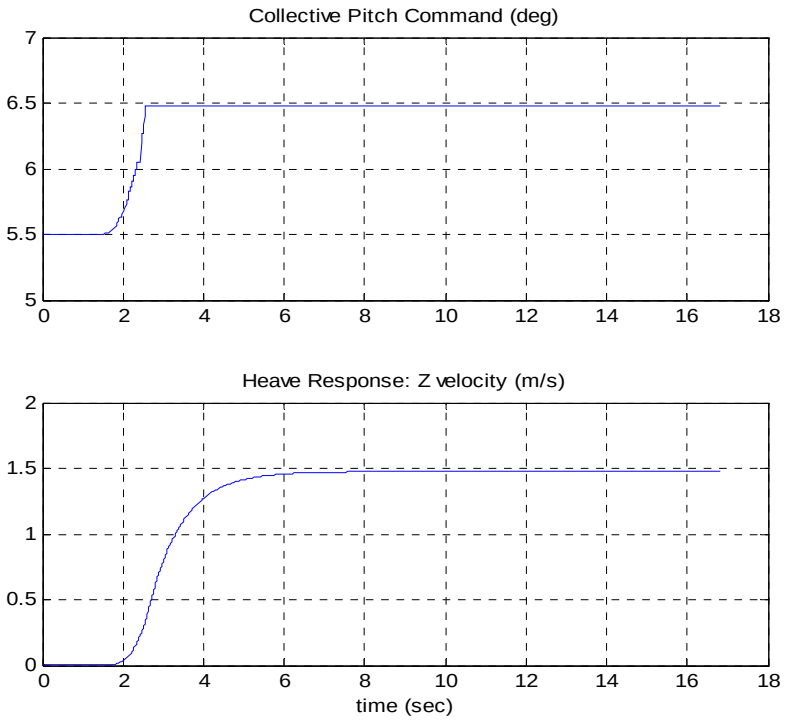


Figure 14. Heave Response to Collective Command Simulation

Figure 14 illustrates that since the thrust returns to Thrust=80.41 N, after the peak, there is no net acceleration, and this is why the velocity settles to a constant value. Equation (3.1) in the helicopter model is implemented in "UpdateMR.m," outlined in Appendix B.

4. Main Rotor Thrust and Inflow Response in Pure Descent

There exist several operating conditions within the descending flight configuration. These are low rate descent, Vortex Ring State (VRS), and the Windmill Break State (WBS) (also known as autorotation). In low rate of descent, there is a well-defined slipstream in the downward direction and, in this case, momentum theory can be applied. In this flight regime, the induced velocity is slightly increasing with increasing descent velocity. This region of allowable low rate descent, in the absence of forward airspeed, is small. In the WBS, the slipstream is in the upward direction and is a condition where the energy imparted onto the rotors by the slipstream is sufficient to drive the rotor. The slipstream is well defined and, momentum theory can also be applied in this state.

Once the descent velocity breaks the limit afforded by momentum theory, the vehicle enters what is known as the Vortex Ring State (VRS). Padfield explains that [4, p. 102]:

> Flight test experience on a small tandem rotor helicopter…reported that the characteristic vibration [of VRS] began at a rate of descent equal to about 23% of the hovering induced velocity and persisted until the rate of descent exceeded 125% of the hover-induced velocity.

This boundary cannot be known a-priori and can only be established during flight-testing. This being said, the model will not be able to accurately simulate vehicle dynamics once the VRS is entered into, and will therefore be avoided during simulations. Padfield's referenced flight test is used to set an initial VRS boundary for the X-Cell model. A more conservative descent limit of 20% hovering induced velocity gives a VRS upper boundary of $V_D = .834$ m/s. The lower VRS boundary is $V_D = 5.2$ m/s. Model simulation is not concerned with aerobatic maneuvers; therefore, a pure descent command will not exceed .834 m/s. It should be mentioned at this point that VRS can be avoided entirely, within the VRS descent boundary, by providing supplemental forward airspeed commands.

In order to generate a smooth simulation, Padfield's linear approximation is used in the VRS region. These are given as [4, p. 118], [5, p. 57]:

$$\lambda_i = \lambda_{ih}\left(1-\frac{\mu_z}{\lambda_{ih}}\right) \quad 0 \le \mu_z \le -.20\lambda_{ih} \tag{3.20}$$

$$\lambda_i = \lambda_{ih}\left(7-3\frac{\mu_z}{\lambda_{ih}}\right) \quad -.20\lambda_{ih} \le \mu_z \le -2\lambda_{ih} \tag{3.21}$$

$$\lambda_i = -\frac{\mu_z}{2} - \sqrt{\frac{1}{4}\mu_z^2 - \lambda_{ih}^2} \qquad \mu_z < -2\lambda_{ih} \tag{3.22}$$

Equation (3.20) describes the main rotor inflow when the vehicle is within the upper descent boundary, between hover and -.834 m/s. Equation (3.21) describes the main rotor inflow when the vehicle is descending at a rate between -.834 m/s and -8.34 m/s. Equation (3.22) describes the main rotor inflow when the vehicle is descending faster than -8.34 m/s.

5. Main Rotor Inflow with Increasing Airspeed

Figure 15 illustrates that for small tip path plane angles $(\alpha = 0)$, where the drag is minimal, the convergent inflow velocity decreases with increased airspeed. As the drag becomes more and more dominant, as is the case with $\alpha = 6°$, the inflow begins to increase again at some critical airspeed.

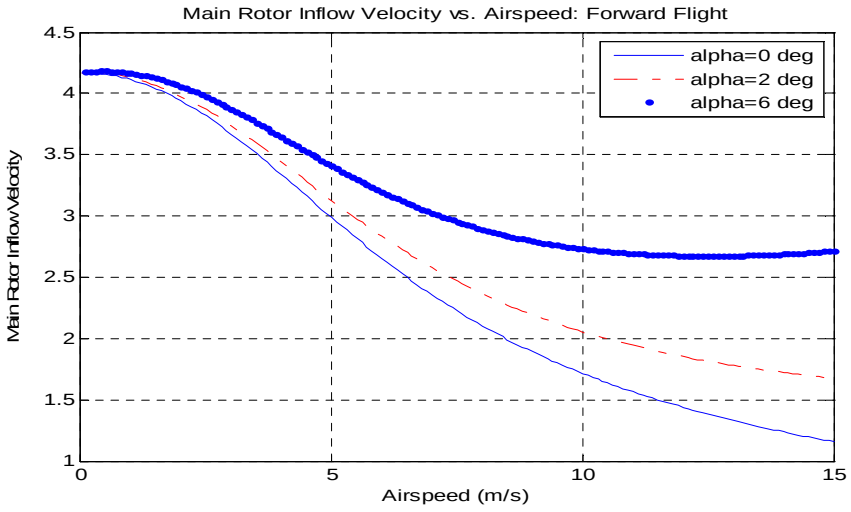


Figure 15. Convergent Inflow Velocities in Forward Flight (After [5])

In most applications, the tip path angle of the main rotor will not be $6°$ when the vehicle is hovering (zero airspeed). In order to compensate for drag, the tip path angle will only increase steadily as the vehicle airspeed increases. Figure 16 is a better depiction of the convergent rotor inflow velocity behavior for the model, over the range of operational airspeeds.

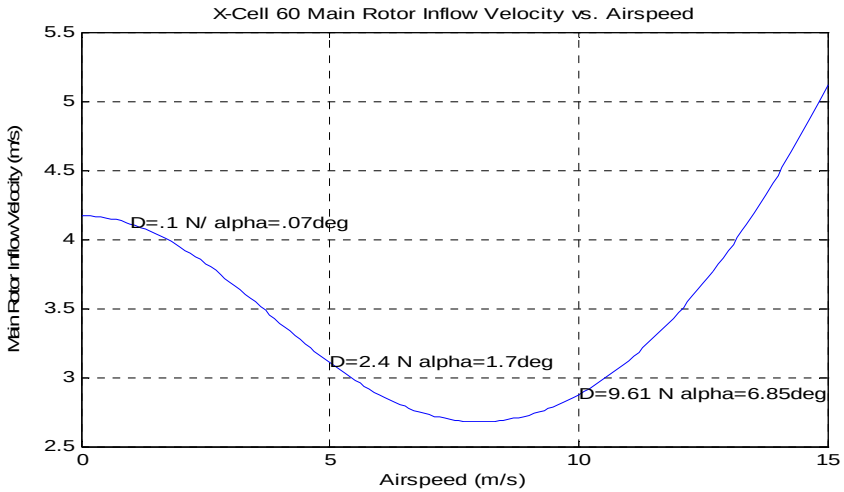


Figure 16. Convergent Inflow Velocities of X-Cel 60 (After [5])

Figure 16 takes into account that the drag and tip path angle increases as the airspeed increases. Figure 16 is labeled in several locations to show the drag and tip path plane angle at that specific operating condition. For example, at a forward airspeed of 1 m/s, the drag is approximately .1 Newtons with a tip path plane angle of $.07°$ and a convergent inflow velocity of 4.11 m/s. At a forward airspeed of 5 m/s, the drag is 2.4 Newtons, the tip path plane angle is $1.7°$ with an inflow velocity of 3.11 m/s. It can also be seen that at approximately 8 m/s, the inflow begins to increase again and gets larger than the hover inflow velocity at around 13.5 m/s (30.2 mi/hr).

Figure 16 also represents the range of main rotor inflow velocities for the vehicle in trimmed flight. That is to say, that the vehicle will generate an inflow of 3.11 m/s, for example, at a steady state trimmed flight of 5 m/s forward airspeed. As the vehicle accelerates to different operating points, there will be inflow transients in between the operating points.

6. Main Rotor Torque

At hover, the coefficient of torque is a combination of the induced torque coefficient and the profile torque coefficient. The induced torque coefficient is due to the [5]:

> …combined effect of the rearward tilt of the incremental lift vectors.

This is a type of aerodynamic drag on the blades, which is proportional to the thrust and is expressed as [3, p. 22]

$$C_{Qi}^{MR} = C_T \sqrt{\frac{C_T}{2}} \tag{3.23}$$

The profile torque can be thought of as coming from friction on the blade surface and is expressed as

$$C_q^{MR} = \frac{c_d^{mr} s}{8} \tag{3.24}$$

Prouty presents where the drag coefficient is treated as a constant,

$$c_d^{mr} = .024$$

Combining both expressions gives the total torque coefficient as [3, p. 22]:

$$(Hover)\ \ C_Q^{MR} = C_T \sqrt{\frac{C_T}{2}} + \frac{c_d s}{8} \tag{3.25}$$

In forward flight, the main rotor torque coefficient is [3, p. 133]:

$$(Forward\ Flight)\ C_Q^{MR} = \frac{s c_d^{mr}}{8}(1+\mu^2) \tag{3.26}$$

The yaw moment generated by this torque is given as [3, p. 22]:

$$N_{MR} = \rho(\Omega R)^2 \frac{bcR^2}{s} C_Q^{MR} \tag{3.27}$$

The following is a treatment of the main rotor yaw moments that are generated at various operating conditions:

Yaw Moment at Hover:

$T = 80.4145\ N$

$$C_T = \frac{T}{\rho(\Omega R)^2 A_d} = .0020769$$

$$C_Q^{MR} = C_T\sqrt{\frac{C_T}{2}} + \frac{c_d^{mr} s}{8} = 2.0986\times10^{-4}$$

$$N_{MR} = \rho(\Omega R)^2 \frac{bcR^2}{s} C_Q = 6.297\ Nm$$

Yaw Moment at 3 m/s Forward Flight:

$T = 81.1725\ N$

$$C_T = \frac{T}{\rho(\Omega R)^2 A_d} = .0020964$$

$$C_Q^{MR} = C_T\sqrt{\frac{C_T}{2}} + \frac{c_d^{mr} s}{8} = 2.1081\times10^{-4}$$

$$N_{MR} = \rho(\Omega R)^2 \frac{bcR^2}{s} C_Q = 6.326\ Nm$$

The conclusion from this exercise is that the forward airspeed is not going to have significant change on the rotor torque. The general trend is that the yaw moment will increase as the main rotor thrust increases.

7. Main Rotor Flapping Dynamics

The main rotor flapping dynamics displays a second order, over damped response. Equation (3.28) expresses that the main rotor flapping is determined by the longitudinal and lateral flap angle commands, $\beta^c{}_{lon}$ and $\beta^c{}_{lat}$, respectively. These commanded flap angles are proportional to the commanded inputs u_{lon} and u_{lat}. The direct relation between u and β is discussed in Chapter III.A. Additionally, roll and pitch motion from the UAV affects the main rotor flapping directly. Mettler gives the linear flapping dynamics as [2, p. 81]:

$$\begin{bmatrix} \dot{\beta}_{lon} \\ \dot{\beta}_{lat} \end{bmatrix} = \begin{bmatrix} -\frac{1}{\tau_f} & 0 \\ 0 & -\frac{1}{\tau_f} \end{bmatrix} \begin{bmatrix} \beta_{lon} \\ \beta_{lat} \end{bmatrix} + \begin{bmatrix} 0 & -1 & \frac{1}{\tau_f} & 0 \\ -1 & 0 & 0 & \frac{1}{\tau_f} \end{bmatrix} \begin{bmatrix} p \\ q \\ \beta^c{}_{lon} \\ \beta^c{}_{lat} \end{bmatrix} \tag{3.28}$$

where τ_f is the main rotor time constant to be determined in system identification. The value of $\tau_f = .1$ seconds was obtained from [2] at hovering conditions. The eigenvalues of the state transition matrix A in (3.28) gives two stable eigenvalues/poles at -10. Taking the A and B matrices and defining the output C matrix as C=[1 0] with input coupling D=[0 0 0 0], the SISO transfer function for this linear system is determined using the following Matlab commands:

```
>> [num,den]=ss2tf(a,b,c,d,3)
>>sys=tf(num,den)
```

The 3 in the argument of the first function defines the input as β^c_{lon}, which is the third input for a four input system. The transfer function representing Equation (3.28) from step command β^c_{lon} to flap angle β_{lon} is then given as:

$$\frac{\beta_{lon}}{\beta^c_{lon}} = \frac{10s + 100}{s^2 + 20s + 100} \tag{3.29}$$

An approximation to the settling time for this second order system is

$$T_s = \frac{4}{\xi\omega_n} \tag{3.30}$$

where the denominator of Equation (3.29) gives

$$s^2 + 2\xi\omega_n + \omega_n^2 = s^2 + 20s + 100 \tag{3.31}$$

Solving for the natural frequency and damping ratio gives settling time estimate of $T_s = .4$ seconds, which is in agreement with the responses illustrated in Figures 16 and 17.

Figures 16 and 17 illustrate the main rotor flapping responses to longitudinal and lateral input commands as well as the vehicle velocity responses in the lateral and longitudinal directions. The lateral step response of the velocity in Figure 17 makes sense for the first 2.5 seconds. The drop in velocity after 2.5 seconds is due to the tendency of the tail boom to rotate away from the direction of lateral turn, in the absence of a tail rotor correction command.

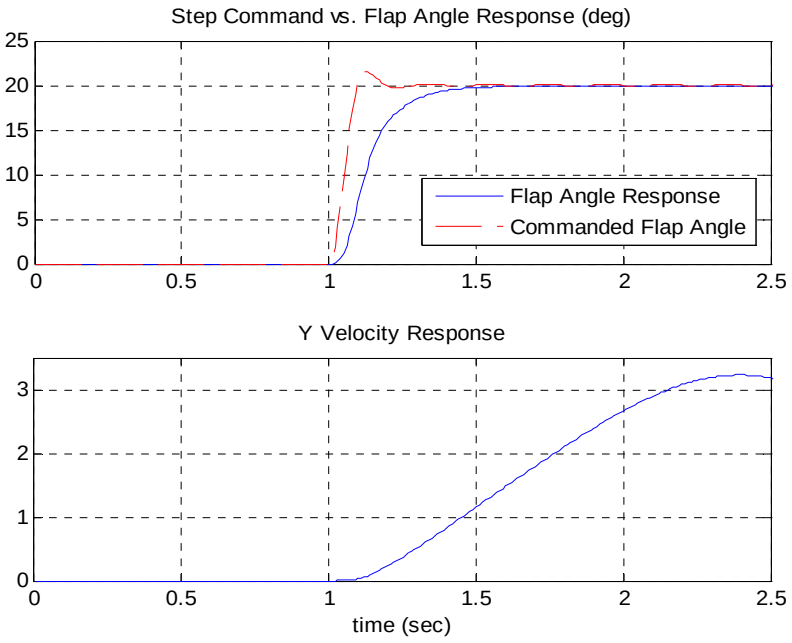


Figure 17. Lateral Flap Angle Response

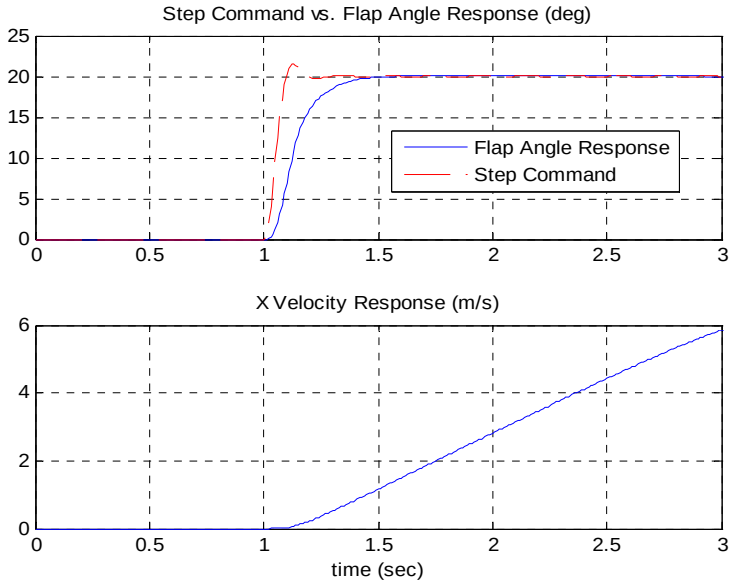


Figure 18. Longitudinal Flap Angle Response

The determination of the velocity responses illustrated in Figures 16 and 17 originate from the thrust vector being translated into the x and y axes as the main rotor flaps. This will be discussed in the next section.

8. Main Rotor Forces and Moments

In order to better appreciate and understand the forces and moments acting on a helicopter, a free body diagram is illustrated in Figure 19.

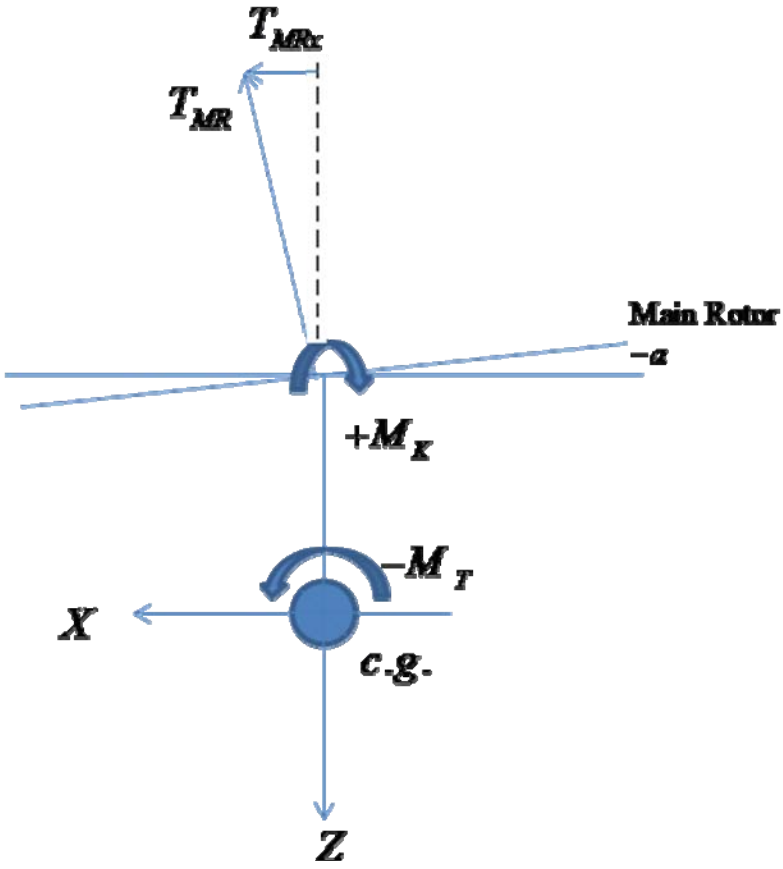


Figure 19. Main Rotor Forces and Moments (After [2, p. 75])

Figure 19 illustrates the forces and moments acting on the helicopter as the main rotor tilts from a flap angle command. As the operator inputs a forward cyclic command, the helicopter main rotor will provide a forward tilt, which generates the longitudinal flap angle $-\beta_{lon}$, illustrated as $(-a)$ in Figure 19. As the main rotor tilts forward, the main rotor thrust vector (T_{MR}) also tilts, thereby generating a force in the +x direction (T_{MRx}).

The thrust (T_{MRx}) then causes an acceleration in the +x direction. Additionally, the x component of the thrust vector acts on the moment arm between the main rotor hub and the vehicle center of gravity, which in turn generates the aerodynamic pitching moment $-M_T$.

As the main rotor tilts, there is a restoring moment due to the centrifugal forces generated by the rotating blades. For now, the restoring moment that is generated from the main rotor centrifugal force is modeled as a spring moment (M_K); this spring moment is modeled as being proportional to the flap angle β_{lon}. An additional moment, not illustrated in Figure 19, is a damping moment which is proportional to any pitching or rolling rate, q or p. This damping moment is physically provided by the stabilizer bar, which behaves as a main rotor cyclic control augmentation device. The main rotor stabilizer bar does not generate any lift; instead, it is a type of gyroscope in the sense that it maintains a horizontal orientation given any main rotor orientation. As the main rotor flaps in any direction, the stabilizer bar maintains a horizontal attitude. Linkages between the stabilizer bar and main rotor control surfaces cause the augmented control inputs from the stabilizer bar to correct for any roll or pitch rates in order to drive the rates to zero. The pitch damping moment, for example, is proportional to the pitch rate q and is expressed as [2]:

$$M_{damp} = -M_a \tau_f q \tag{3.32}$$

where M_a and τ_f are constants to be determined by system identification. The M_a term describes the rate at which the vehicle pitches given a perturbation in the flap angle β_{lon}. See Appendix A for identification model parameters used from the MIT Instrumented X-Cell 60. The X-Cell 60 model parameters at hover are used to model the unknown T-Rex Align parameters until system identification is completed.

Conducting a force and moment analysis for all 6-DOF gives Equations (3.33) and (3.34), for the summation of all forces and moments.

$$
\begin{aligned}
X_{MR} &= T_{MR}\sin(\beta_{lon})\cos(\beta_{lat}) \\
Y_{MR} &= T_{MR}\sin(\beta_{lat})\cos(\beta_{lon}) \\
Z_{MR} &= T_{MR}\cos(\beta_{lon})\cos(\beta_{lat})
\end{aligned}
\tag{3.33}
$$

$$
\begin{aligned}
L_{MR} &= Y_{MR}h_{MR} + (K_{\beta})\beta_{lat} - L_b\tau_f p \\
M_{MR} &= X_{MR}h_{MR} + (K_{\beta})\beta_{lon} - M_a\tau_f q \\
N_{MR} &= \rho(\Omega R)^2 \frac{bcR^2}{s} C_Q
\end{aligned}
\tag{3.34}
$$

It should be noted that the Simulink model implementation of Equation (3.34) takes into consideration the sign of the flap angle before applying the equation. That is, if the longitudinal flap angle (β_{lon}) is positive, the restorative moment $(K_{\beta})\times\beta_{lon}$ must be negative; if the flap angle (β_{lon}) is negative, the restorative moment $(K_{\beta})\times\beta_{lon}$ must be positive. This ensures that the aerodynamic pitching moment and the restorative moments are opposite in sign and will cancel each other. Note that the yaw axis main rotor moment (N_{MR}) was covered in Chapter III.B.6.

9. Pitch/Roll Rate and Velocity Response to Main Rotor Forces and Moments

Figure 20 illustrates the pitch and velocity response generated by applying equations (3.33) and (3.34) to a lateral right-stick cyclic input. These responses are in close agreement with the full-scale helicopter response in [4, p. 349].

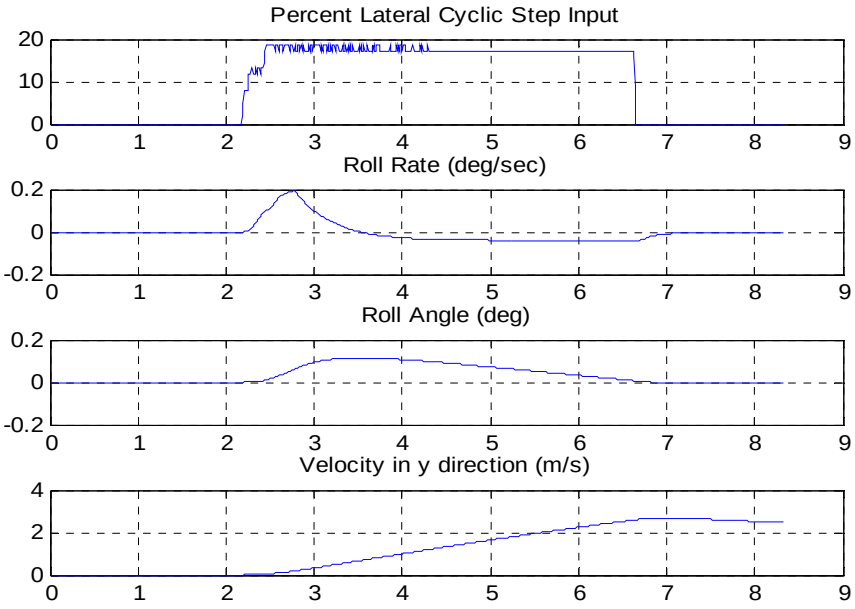


Figure 20. Lateral Cyclic Step Response

Figure 20 illustrates that a step input in lateral cyclic generates an increasing roll rate. The difference between the model roll rate response and the full scale helicopter response of [4] is that the model response is more stable in that the roll rate damps back to zero rate while the cyclic input is still active. This is highly desirable and characteristic of the miniature helicopter model. If the roll rate is too damped, the roll derivative L_b can be tuned, further, this will be determined in future system identification of the T-Rex Align. Figure 20 also illustrates that the roll angle only attains a maximum of $.2°$ roll with a damping back to zero roll angle. This is also highly desirable because it allows the vehicle to continue to increase its velocity in the y direction without being hindered by increasingly large roll angles generated by the rolling moment. This illustrates that the restorative and damping moments do a good job of canceling aerodynamic moments generated by the flapping dynamics. Figure 20 also shows that the velocity in the y direction increases as the cyclic input is maintained at the 20% position. The longitudinal responses are similar to those illustrated in Figure 20.

C. TAIL ROTOR DESIGN

The tail rotor behaves much the same way the main rotor does. Therefore, the tail rotor thrust and inflow are first determined in the hover conditions. A tail rotor collective command will then cause the tail rotor thrust to either increase or decrease, depending on the direction of the command. The rotor inflow will either increase with increased tail rotor trust or decrease with decreased tail rotor thrust. As the tail rotor inflow velocity increases with increased thrust, for example, the tail rotor thrust will first peak then it will begin to decrease just as was the case with the main rotor. The same iterative procedure will be applied to determine the convergent inflow for the tail rotor. There are several other differences, however, that need to be discussed.

1. Tail Rotor Hub Airspeeds and Main Rotor Wake Considerations

Modeling of the tail rotor requires an understanding of how the helicopter's motion affects the local tail rotor airspeed components. In pure translational motion, the tail rotor hub airspeed is straightforward to calculate. If the vehicle is flying in a pure sideways motion (positive y-axis), the velocity of the tail will be the same as the vehicle sideways airspeed. If the vehicle is experiencing roll, pitch, or yaw rates, this will affect the tail rotor velocity directly. The tail rotor hub airspeed in the y direction is given by Gavrilets as [1, p. 51]:

$$v_{tr} = v_a - l_{tr} r - h_{tr} |p| \tag{3.35}$$

where v_a is the resultant airspeed of the vehicle in the y direction, compensated for wind disturbance. Equation (3.35) expresses the tail rotor hub velocity as a combination of the translational velocity of the tail (v_a), angular velocity of the tail due to tail rotor yaw ($l_{tr} r$), and angular velocity of the tail due to roll ($h_{tr} p$). If the vehicle travels in the positive y direction, a positive yaw rate (clockwise) will reduce the net tail rotor airspeed in the y direction. If the yaw rate is negative (counterclockwise, with $-r$), the tail rotor airspeed becomes $v_{tr} = v_a + l_{tr} r$ meaning that the tail rotor hub airspeed is faster than the vehicle center of gravity. The tail rotor will then begin to overshoot the vehicle center of

gravity. It has been taken into account that this is a tractor type tail rotor (starboard side rotor). The tractor type rotor's thrust is in the opposite direction of the vertical fin and the tail rotor inflow stream flows towards the tail fin.

With respect to the roll rate contribution to the hub airspeed in the y direction, the absolute value of the roll rate is necessary. As the vehicle travels in the positive y direction, a roll rate in either direction (port or starboard roll) will contribute a component of airspeed in the negative y direction because of the tail rotor position. If the vehicle travels in the negative y direction, the roll component will have the effect of increasing the tail rotor hub airspeed regardless of the direction of the roll.

The normalized rotor inflow, normal to the tail rotor is [1]:

$$\mu_{ztr} = \frac{v_{tr}}{\Omega_{tr} R_{tr}} \tag{3.36}$$

The tail rotor is approximated as 4.46 times faster than the main rotor speed ($\Omega_{tr} = 4.46\Omega_{MR}$) and is calculated as $\Omega_{tr} = 778.2\ rad/\sec$. The tail rotor hub airspeed in the z direction is given as:

$$w_{tr} = w_a - l_{tr} q + K_\lambda V_{imr} \tag{3.37}$$

where w_a is the vehicle airspeed in the z direction, compensated for wind disturbance, and ($l_{tr} q$) is the angular airspeed component due to a pitching rate. The term K_λ is a wake intensity factor that increases as the tail rotor becomes more and more immersed in the main rotor wake. The rotor wake has the effect of increasing the airspeed of the tail rotor hub, in the z direction. The V_{imr} term is the main rotor induced velocity. In low forward airspeed and hovering flight, the wake intensity factor $K_\lambda = 0$ and grows as the tail rotor becomes more and more immersed in the main rotor wake as the forward airspeed increases. The condition for zero wake intensity (near hovering flight) is given by [1]:

$$\frac{u_a}{V_{imr} - w_a} \le g_i \quad (K_\lambda = 0) \tag{3.38}$$

The tail rotor is fully in the wake when [1]:

$$\frac{u_a}{V_{imr} - w_a} \geq g_f \quad (K_\lambda = 1.5) \tag{3.39}$$

in which case, $K_\lambda = 1.5$. The g_i and g_f terms are determined from the vehicle geometry and are given by [1]:

$$g_i = \frac{l_{tr} - R_{mr} - R_{tr}}{h_{tr}} = .0625 \tag{3.40}$$

$$g_f = \frac{l_{tr} - R_{mr} + R_{tr}}{h_{tr}} = 3.3125 \tag{3.41}$$

Equations (3.38) through (3.41) say that the main rotor wake does not impinge on the tail rotor airflow until the vehicle forward airspeed is greater than 6.25% of the resultant main rotor inflow.

Gavrilets assumes a linear growth of the wake intensity with increasing forward speed. The wake intensity is given by [1]:

$$K_\lambda = 1.5 \times \frac{\frac{u_a}{V_{imr} - w_a} - g_i}{g_f - g_i} \quad for \ \frac{u_a}{V_{imr} - w_a} > g_i \tag{3.42}$$

and Equation (3.37) can be rewritten as:

$$w_{tr} = w_a - l_{tr} q + 1.5 \left(\frac{\frac{u_a}{V_{imr} - w_a} - .0625}{3.25} \right) V_{imr} \tag{3.43}$$

The tail rotor hub airspeed in the x direction is given by:

$$u_{tr} = u_a + \left| l_{tr} q + l_{tr} r \right| \tag{3.44}$$

Equation (3.44) states that the tail rotor hub airspeed in the x direction is a combination of the vehicle airspeed u and the pitch and yaw rates. Additionally, it states that the pitch and yaw rates contribute positive airspeed components, no matter the sign of the pitch and yaw rate. For example, if the vehicle is traveling in the positive x direction, a pitch rate in either direction (up or down) will contribute a positive airspeed component to the tail rotor hub, in the x direction. If the vehicle travels in the negative x direction, the positive contribution of the pitch or yaw rate will make the tail rotor airspeed less negative and the tail rotor begins to catch up to the vehicle center of gravity.

With all of the tail rotor airspeed components defined, the tail rotor hub airspeed magnitude is then:

$$V_{tr} = \sqrt{u_{tr}^2 + v_{tr}^2 + w_{tr}^2} \tag{3.45}$$

The tail rotor advance ratio is expressed as [1]:

$$\mu_{tr} = \frac{\sqrt{u_{tr}^2 + w_{tr}^2}}{\Omega_{tr} R_{tr}} \tag{3.46}$$

2. Tail Rotor Inflow and Thrust

To determine the tail rotor inflow, Equation (3.11) is modified to suit the tail rotor conditions. Equation (3.11) shows that the main rotor inflow is the sum of the inflow plus any normal airspeed component, which is given by the $V_\infty \sin(\alpha)$ term. In the tail rotor case, the tail rotor tip path plane will not be canted because there are no modeled cyclics. Therefore, the airspeed normal to the tail rotor blades will be given simply by Equation (3.35). Equation (3.11) now becomes:

$$\begin{aligned} \lambda^{tr} &= \frac{v_{tr}}{\Omega R} + \frac{v_i}{\Omega R} \\ \lambda^{tr} &= \mu_{ztr} + \lambda_i \end{aligned} \tag{3.47}$$

Substituting the rotor inflow expression, equation (3.8), into (3.47) gives the tail rotor inflow as:

$$\lambda^{tr} = \mu_{ztr} + \frac{C_T}{2\sqrt{\mu_{tr}^2 + \lambda^2}} \tag{3.48}$$

The Newton-Raphson iteration scheme for the tail rotor is:

$$\lambda_{n+1} = \lambda_n - \left[\frac{f(\lambda)}{f'(\lambda)}\right]_n \tag{3.49}$$

$$f(\lambda) = \lambda - \mu_{ztr} - \frac{C_T}{2\sqrt{\mu_{tr}{}^2 + \lambda^2}} \tag{3.50}$$

$$f'(\lambda) = 1 + \frac{C_T}{2}\left(\mu^2 + \lambda^2\right)^{-3/2}\lambda \tag{3.51}$$

The tail rotor inflow is determined iteratively, as with the main rotor, using Equations (3.49) through (3.51). The initial estimate (λ_o) is the normalized tail rotor inflow at hover. The tail rotor inflow behaves just as the main rotor inflow in that it follows the thrust trend. If the tail rotor thrust increases, the tail rotor inflow increases. As the inflow increases, the tail rotor thrust will settle at a new equilibrium after it peaks. This is more obvious when the tail rotor thrust is expressed as:

$$\begin{aligned} T_{T(tr)} &= \frac{\rho\left(\Omega_{tr}R_{tr}\right)^2 \pi\left(R_{tr}\right)^2 as}{2}\left[\delta_{ped}\left(\frac{1}{3} + \frac{\mu_{tr}^2}{2}\right) - \left(\frac{\mu_{ztr}}{2} + \frac{\lambda}{2}\right)\right] \\ T_{T(tr)} &= 192.93\rho\left[\delta_{ped}\left(\frac{1}{3} + \frac{\mu_{tr}^2}{2}\right) - \left(\frac{\mu_{ztr}}{2} + \frac{\lambda}{2}\right)\right] \end{aligned} \tag{3.52}$$

When a right pedal command is given $\left(\delta_{ped}\ is\ positive\right)$, using the previous value of the inflow $\left(\lambda \text{ at hover}\right)$ will cause the thrust to increase. The new value of the tail rotor is fed into the Netwon-Raphson technique and gives the new value of the tail rotor

inflow, which is slightly larger than the previous value. The tail rotor inflow will continue to increase until the rudder input has settled. Once the rudder input settles, the first term in (3.52) settles as the inflow continues to increase. This causes the tail rotor thrust to settle after it attains a peak value. Figure 21 illustrates the tail rotor thrust and inflow behavior to a step in tail rotor collective:

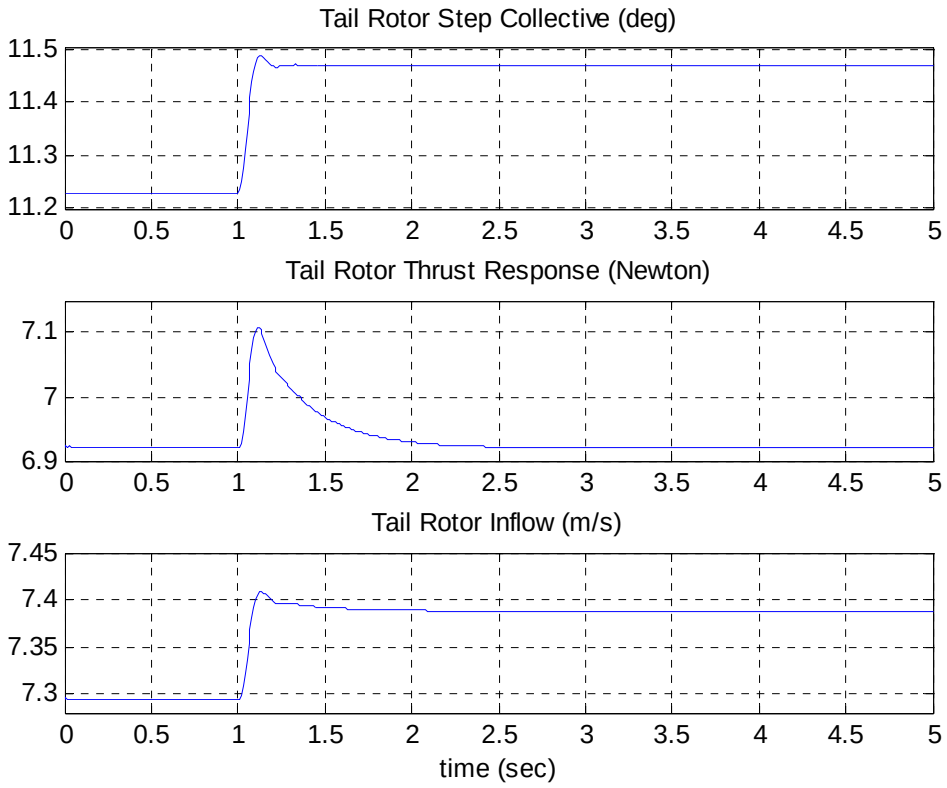


Figure 21. Tail Rotor Thrust and Inflow Responses to Step Collective

The tail rotor thrust behaves much like the main rotor in that a step collective input yields the characteristic thrust response with an initial rise, followed by a damping back to the equilibrium value.

It has been demonstrated that the torque produced by the main rotor is $N_{MR} = 6.297\ Nm$. Normalizing this by the tail arm ($l_{tr} = .91\ m$) gives the force that the tail rotor needs to generate to counteract the main rotor torque. This gives:

$$Y_{tr(hov)} = 6.92\ N$$

Since the main rotor torque produces a positive yaw (clockwise rotation of the vehicle as viewed from above), the tail rotor force at hover ($Y_{tr(hov)}$) needs to point in the positive y direction in order to counteract the main rotor torque.

The tail rotor coefficient of thrust can be determined for any other flight configuration from the airspeed and angular rate conditions. Once the tail rotor inflow is determined, the tail rotor thrust can be derived. Padfield gives the tail rotor thrust as [4, p. 141]:

$$Y_{tr} = \rho\left(\Omega_{tr} R_{tr}\right)^2 \left(\pi R_{tr}^2\right) C_{T(tr)}\, f_T \tag{3.53}$$

where the coefficient of the tail rotor thrust is [4, p. 142]:

$$C_T = \frac{T_{tr}}{\rho\left(\Omega_{tr} R_{tr}\right)^2 \pi\left(R_{tr}\right)^2} \tag{3.54}$$

Substituting Equation (3.54) into (3.53) gives:

$$Y_{tr} = T_{tr}\, f_T \tag{3.55}$$

The blockage factor f_T accounts for thrust losses in pusher type tail rotors. Since this is a tractor type tail rotor, the blockage factor will be ignored and Equation (3.55) reduces to:

$$Y_{tr} = T_{tr} \tag{3.56}$$

The net tail rotor force needed to counteract the main rotor torque is $Y_{tr(hov)} = 6.92\ N$, which is the same as the thrust required and gives a coefficient of thrust $C_{T(tr)}^{hov} = 10.3954 \times 10^{-3}$.

At hover, the normal airflow velocity is equal to the tail rotor inflow velocity and Equation (3.8) becomes $T = 2\rho A_{d(tr)} v_i^2$ and solving for the induced velocity at hover gives:

$$v_{i(tr)}^{hov} = 7.294\ m/s$$

$$\lambda_{i(tr)}^{hov} = \frac{v_{i(tr)}^{hov}}{\Omega_{tr} R_{tr}} = 72.095 \times 10^{-3}$$

Solving Equation (3.52) for the tail rotor collective required at hover gives Equation (3.57):

$$\delta_{ped} = \frac{\left[\frac{2C_{T(tr)}}{as} + \frac{\lambda}{2} - \frac{\mu_{ztr}}{2} \right]}{\left(\frac{1}{3} + \frac{\mu_{tr}^2}{2} \right)} \tag{3.57}$$

Solving for the trim tail rotor collective at hover gives

$$\delta_{ped}^{hov} = 3\left[\frac{2C_{T(tr)}}{as} + \frac{\lambda}{2} \right]$$

$$\delta_{ped}^{hov} = .19598\ rad\ \ (11.23°)$$

Tail rotor collective ranges used for this model will be -15˚ to +25˚ with a zero collective input that generates the trim tail rotor collective of 11.23˚.

The tail rotor forces and moments are:

$$\begin{aligned} Y_{tr} &= T_{tr} \\ N_{tr} &= Y_{tr} l_{tr} \end{aligned} \tag{3.58}$$

3. Yaw Response to Tail Rotor Forces and Moments

Figure 22 illustrates the yaw step response when applying the tail rotor thrust and moments described in Section 2.

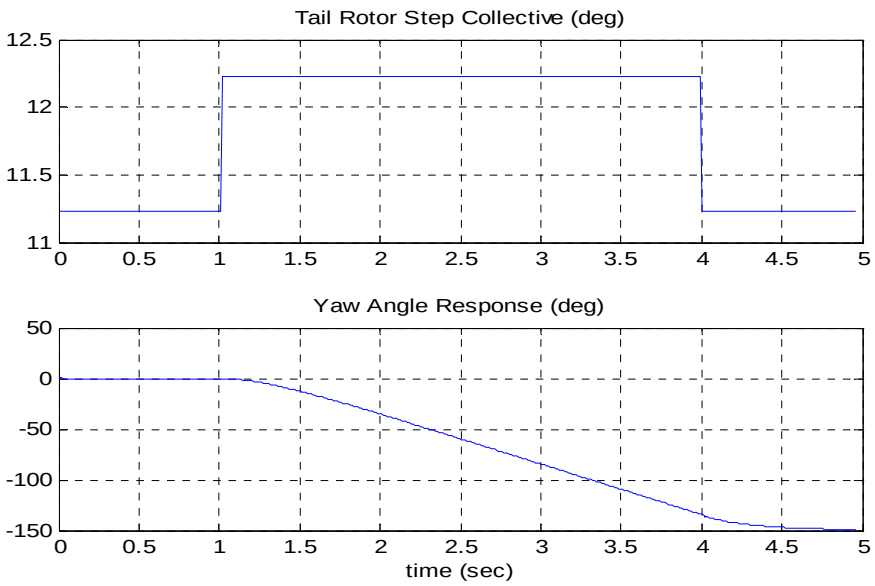


Figure 22. Tail Rotor Thrust and Inflow Responses to Step Collective

The yaw angle step response is in agreement with the yaw response of [1, p. 112], as expected. The yaw angle increases in a counter-clockwise direction with right pedal input and damps down to almost zero yaw rate after the pedal input is taken out. The slight drift is accounted for by the momentum that has already been imparted to the tail.

D. AERODYNAMIC FORCES

There are two primary aerodynamic forces acting on the RW UAV. They are drag and the aerodynamic forces generated by the main rotor airflow around the UAV body.

1. Determination of Drag and Main Rotor Tip-Path Plane Angle

As the vehicle approaches forward airspeeds comparable to the hover induced velocity (4 m/s), drag begins to increase to the extent that it has to be compensated for. Prouty explains that the tip path plane angle develops as a direct result of the drag forces that act on the vehicle. For a vehicle in forward flight, the drag force acts in the negative x direction. In order to balance the forces in the x-direction, a longitudinal cyclic command must be put in. This longitudinal cyclic input causes the main rotor disk to tilt by an angle (α) with respect to the horizontal plane. As the disk tilts, the thrust vector

also tilts (thrust vector is normal to the disk) and creates a thrust component in the positive x direction that will counter balance the drag force.

The sum total of all drag forces on the vehicle is known as parasite drag and is expressed in terms of an equivalent flat plate area, f. According to Prouty [3, p. 132]:

> …the equivalent flat plate area is the frontal area of a flat plate with a drag coefficient of 1, which has the same drag as the object whose drag is being estimated.

Table 4 is a list of vehicle components for Prouty's example helicopter along with initial estimates for the T-Rex Align. The baseline for estimates of the T-Rex Align is the fuselage flat plate area. Once the flat plate area for the T-Rex Align is estimated, the conversion factor used to map from the model to Prouty's example is determined and used for all the other components. The dimensions used for estimating the vehicle fuselage flat plate area are (.24x.24) m, this gives a conversion factor of 9.93×10^{-3}. Components that contribute very little parasite drag are omitted
[3, p. 132]:

Component (Prouty's Example)	**Flat Plate Area (ft^2)**	**X-Cell 60 Estimate (m^2)**
Fuselage	5.8	.05760
Main Rotor Hub and Shaft	7.0	.06952
Main Landing Gear	1.2	.01192
Rotor-Fuselage Interference	1.3	.01291
Miscellaneous	.5	$4.9655\text{x}10^{-3}$
Total	15.8	.1569

Table 4. Component-Wise Flat Plate Area Estimates

The total parasite drag is proportional to the airspeed and is expressed as [3, p. 132]:

$$D_p = f\frac{\rho}{2}V^2 \ (N) \tag{3.59}$$

The main rotor tip-path plane is approximated as [3, p. 133]:

$$\alpha = -57.3\frac{D}{G.W.} \ (\text{deg}) \tag{3.60}$$

This angle is the main rotor tip path plane angle, with respect to the horizontal plane, that is required to keep the vehicle in trim flight, for a given forward velocity. It will be shown later, how the fuselage dynamics affect this angle as well as how it relates to the pilot cyclic input. Since the thrust vector is normal to the tip-path plane, it follows that the thrust in the z-direction, Z_{MR}, is:

$$Z_{MR} = T\cos(\alpha) \quad (N) \tag{3.61}$$

The force in the x-direction due to the main rotor is:

$$X_{MR} = T\sin(\alpha) \quad (N) \tag{3.62}$$

Balancing forces in the x direction gives:

$$T\sin(\alpha) = f\frac{\rho}{2}V^2 \tag{3.63}$$

Solving for the trim thrust required in this flight configuration gives:

$$\begin{aligned} &(Forward\ Flight: V > v_i) \\ &T = \frac{f\frac{\rho}{2}V^2}{\sin(\alpha)} \quad (N) \\ &v_i = \frac{v_{ihover}^2}{\sqrt{V^2 + 2Vv_i\sin\alpha + v_i^2}} \end{aligned} \tag{3.64}$$

Again, this is the trim condition thrust required to balance the main rotor forces, the vehicle weight, and the parasite drag.

2. Fuselage Forces

According to Gavrilets, the rotor downwash is deflected by the forward and side velocity, when near the hover flight regime. The deflection of the rotor downwash creates

a force that acts opposite the direction of movement. The fuselage drag forces in the X and Y directions are given by [1, p. 47]

$$X_{fus} = S_x^{fus} \frac{1}{2} \rho v_{1hov} u \tag{3.65}$$

$$Y_{fus} = S_y^{fus} \frac{1}{2} \rho v_{1hov} v \tag{3.66}$$

where S_x^{fus} and S_y^{fus} are projected cross sectional areas of the fuselage on the YZ and XZ planes.

In forward flight (when the translational velocity is greater than the main rotor inflow velocity), the fuselage drag forces are expressed as [3]

$$X_{fus} = S_x^{fus} \frac{1}{2} \rho U_e u \tag{3.67}$$

$$Y_{fus} = S_y^{fus} \frac{1}{2} \rho U_e v \tag{3.68}$$

Where U_e is the trim airspeed. Gavrilets [1] gives the tail rotor hub velocity magnitude as:

$$V_\infty = \sqrt{u_a^2 + v_a^2 + (w_a + v_{ihov})^2} \tag{3.69}$$

and Equations (3.67) and (3.68) can be expressed as [1]:

$$X_{fus} = -\frac{1}{2} \rho S_x^{fus} u_a V_\infty \tag{3.70}$$

$$Y_{fus} = -\frac{1}{2} \rho S_y^{fus} v_a V_\infty \tag{3.71}$$

The fuselage drag force in the Z direction is expressed as [1]:

$$Z_{fus} = -\frac{1}{2} \rho S_z^{fus} (w_a + v_{ihov}) V_\infty \tag{3.72}$$

Mettler assumes that $S_y^{fus} \approx 2.2 S_x^{fus}$, $S_z^{fus} \approx 1.5 S_x^{fus}$, where S_x^{fus} is determined by the vehicle cross sectional area with respect to the YZ axis, when the vehicle was operating

at 15.4 m/s forward air speed. This resulted in $S_x^{fus} = .1\, m^2$. This initial estimate will also be used for modeling T-Rex Align UAV. The remaining cross sectional areas are:

$$S_x^{fus} = .1\, m^2$$
$$S_y^{fus} = .22\, m^2$$
$$S_z^{fus} = .15\, m^2$$

In summary, the main rotor thrust is calculated by using Equation (3.1), which relates the operator commanded collective (θ_o), the vehicle velocity states, the main rotor aerodynamic and dimensional parameters, and the main rotor inflow (λ_i). The Newton-Raphson iteration technique is used to compute the convergent inflow at a rate of 100 Hz. The main rotor flapping dynamics is modeled as a second order, linear system whose inputs are commanded flap angles, the pitch rate, and roll rate. The tail rotor was modeled like the main rotor, with the exception that there are no tail rotor cyclics. This ensures that the tail rotor thrust always acts normal to the helicopter fin plane. Also, the tail rotor has its own tail rotor hub velocities, which are not necessarily the same as the center of gravity velocity states. The forces and moments on the vehicle center of gravity are computed using Equations (3.33) and (3.34). These forces and moments are summed to provide force and moment components to the 6-DOF block set.

The following chapter will go through the linearization of the nonlinear model in order to uncover the state boundaries for which the controller design will be useful.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. LINEARIZATION OF THE NONLINEAR ROTARY-WING MODEL

The 6-DOF RW UAV model is highly nonlinear. Linearizing the model about a certain operating point makes the control design process simpler. The goal is to obtain the necessary linear models that describe the helicopter dynamics and to design the PID controllers that will minimize the flight trajectory errors during SIL testing.

Matlab has several linearization tools that can be applied to most nonlinear models. The tools that were applied for this linearization task are the functions "linmod" and "n4sid." The linmod function generates a state space model, given the model name, initial state vector, and inputs. The "n4sid" function is used to obtain linear models for dynamics that need to be identified through system identification. Please see Appendices C and D for Matlab script implementation of these functions.

A. TRIMMING OF THE NONLINEAR MODEL

The trim solution is given by the control inputs

$$u_e = [u_{lon}^e \quad u_{lat}^e \quad u_{col}^e \quad u_{ped}^e]$$

required to maintain the helicopter at specified states

$$x_e = \begin{bmatrix} u & v & w & \phi & \theta & \psi & p & q & r \end{bmatrix}$$

Trimmed flight requires that the rate of change (of magnitude) of the aircraft's state vector,

$$\dot{x}_e = \underline{0}$$

and the sum of the forces and moments on the aircraft are zero. Once an operation point or steady states are chosen, the aircraft will be trimmed. That is, the trim inputs required to maintain the aircraft in the specified operating condition will be determined. The trim inputs and states are determined because this allows the use of small perturbation theory to make estimates of the helicopter states about a trim operating point. Small perturbation

theory says that, during perturbed motion, the helicopter behavior can be described as a perturbation from the trim, written as [4, Pg. 208]:

$$x = x_e + \delta x$$

The operation point that was chosen for trimming is the hovering flight condition where all states

$$x_e = \begin{bmatrix} u & v & w & \phi & \theta & \psi & p & q & r \end{bmatrix}$$

are zero. Appendix D outlines the commands given to find the trim settings for any specified set of state conditions. The desired states are specified and the error tolerances are defined. At first, the nonlinear model response is simulated with an initial estimate of the trim inputs required. The state values at the end of the simulation are then taken and compared to the desired values and the errors are computed. The computed errors are then used to adjust the appropriate control input and the simulation is run again. This process is conducted iteratively until the adjusted trim conditions can maintain the desired helicopter states within the defined tolerances.

B. LINEARIZATION OF THE LATERAL AND LONGITUDINAL DYNAMICS

The lateral and longitudinal dynamics are generated by the main rotor flapping response along the y and x axes, respectively. A stick command (left-right) for a lateral flap angle β_{lat}, for example, will cause the lateral cyclic servo to rotate and generate a main rotor flap angle command. This tilting of the rotor in the y direction generates a force in the y direction as well as a rolling moment, which causes the UAV to roll. A stick command (forward-aft) for a longitudinal flap angle β_{lon}, on the other hand, will cause the longitudinal cyclic servo to rotate and generate a main rotor flap angle command in the longitudinal direction (+/-x direction). The tilting of the rotor in the x direction generates a force in the x direction as well as a pitching moment.

Once the hover trim conditions have been determined, as described in section A, the lateral and longitudinal dynamics linear models will be obtained by using the Simulink function “linmod.” The linmod function extracts the state space matrices A, B, C, and D given the equilibrium state and input vectors, x_e and u_e, respectively. Once the

state space representation is derived, the minimum realization, single input single output (SISO) model is extracted. The minimum realization reduces the model order and complexity for states that contribute little to no dynamic behavior for the degree of freedom in question. The details on the commands used to generate the lateral and longitudinal models are in Appendices E and F.

Figure 23 illustrates the 6-DOF nonlinear model with four command inputs and nine state outputs. Figure 24 illustrates more of the model's internal structure.

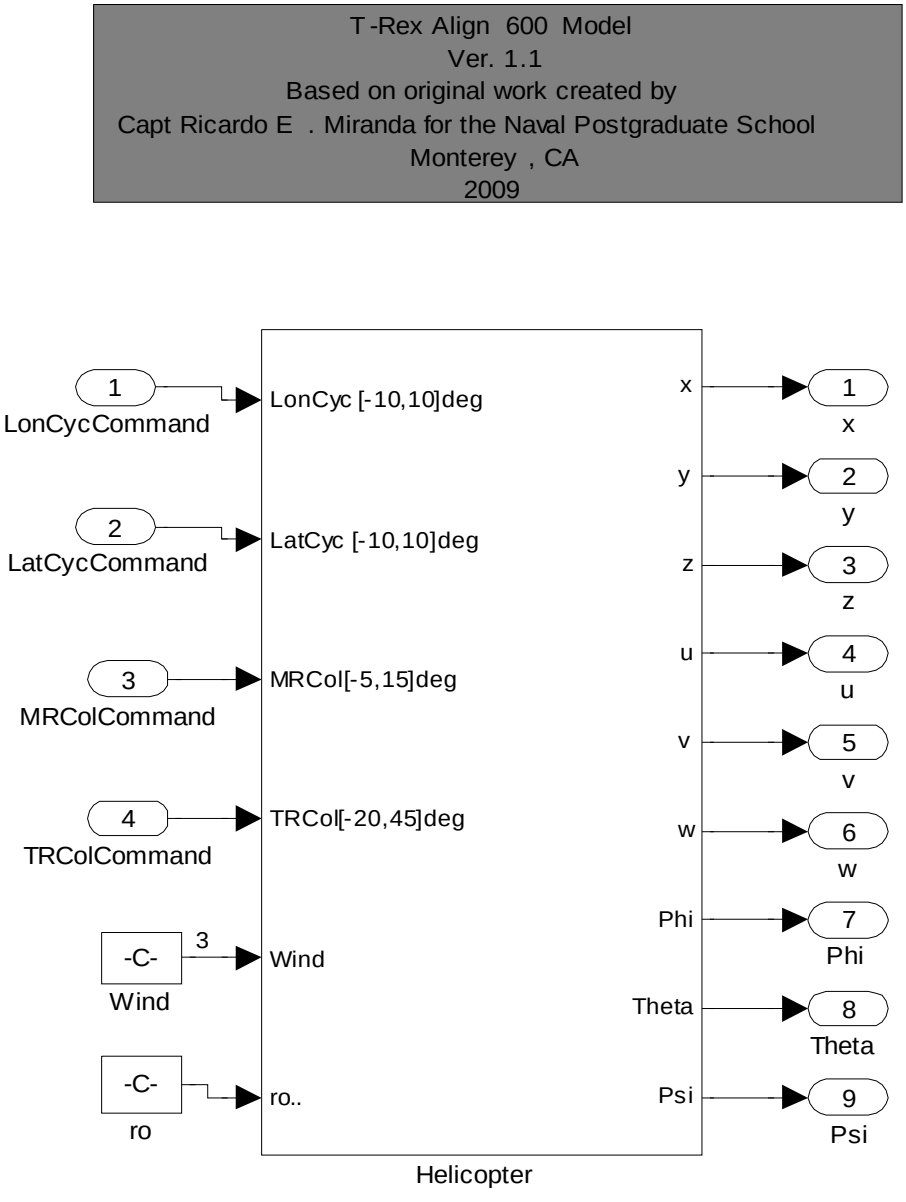


Figure 23. Nonlinear 6-DOF T-Rex Align 600 Model

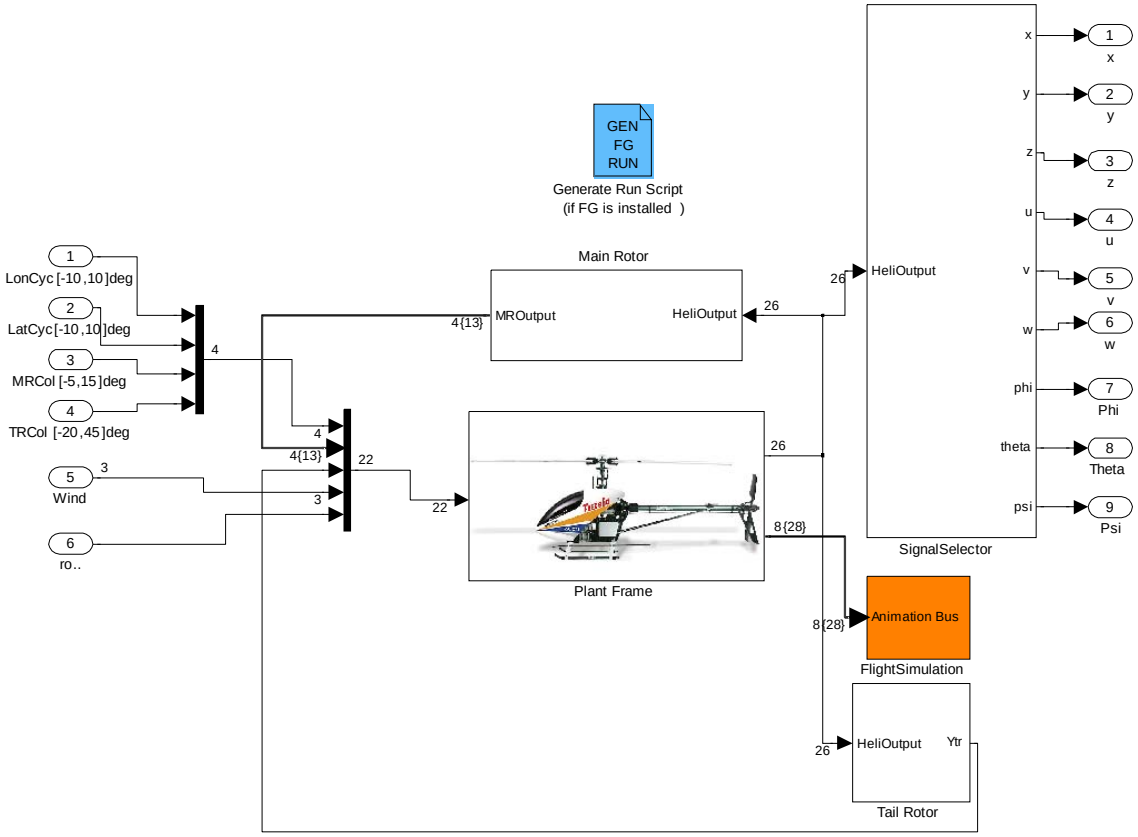


Figure 24. Nonlinear 6-DOF T-Rex Align Model (Internal Structure)

The file “RunModel.m” (Appendix D) is run in order to generate step input responses and to generate linear models for all six degrees of freedom. Once all of the linear models are obtained, the nonlinear step responses are compared to the linear model step responses. A good linear model provides similar responses to the nonlinear model for a short period of time, before the linear response begins to diverge from the nonlinear response. Appendix G illustrates the internal structure of the T-Rex Align 600 linear model.

Figure 25 illustrates the open-loop step response of the nonlinear model to a 2° longitudinal step command in flap angle as well as the linear model’s response.

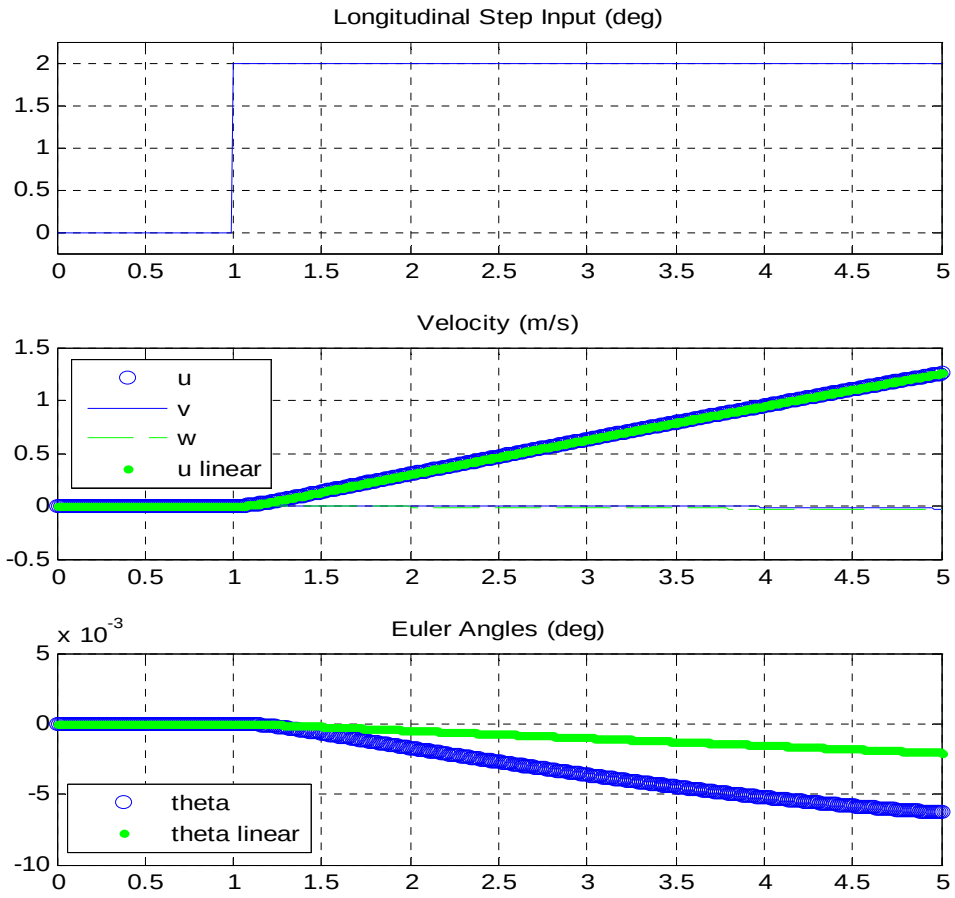


Figure 25. Longitudinal Dynamics / Linear Model Validation

Figure 25 illustrates that the linear model represents the nonlinear longitudinal model quite accurately in the velocity u. The pitch response of the helicopter is not directly controllable and is designed with a high degree of dynamic stability. Controller design is designed around the longitudinal velocity model obtained. Figure 25 also illustrates that the linear model is accurate in representing the surge velocity of the vehicle up through 1.5 m/s. This indicates that the helicopter model can be disturbed from hover, out to 1.5 m/s, without departing from its linear region. This observation makes it possible to design a flight path controller near the hover flight regime. The input to output transfer functions and eigenvalues for the principal dynamics are listed in Appendix H.

Figure 26 illustrates the step response of the nonlinear model to a lateral step command of 2° in flap angle. Also illustrated are the linear model responses.

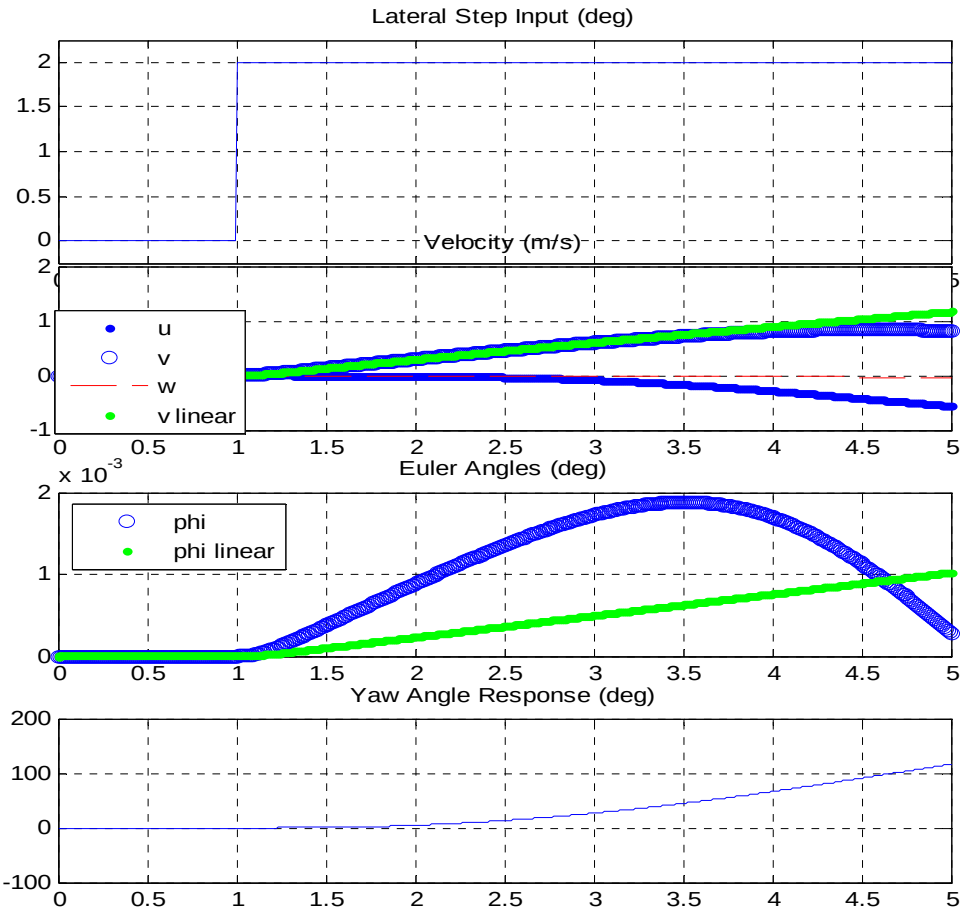


Figure 26. Lateral Dynamics/Linear Model Validation

The vehicle behaves differently when disturbed in the lateral direction, as compared to the longitudinal responses. This is because the helicopter has a tendency to rotate in the direction of lateral disturbance, in the absence of tail rotor thrust compensation. As the vehicle's yaw angle increases beyond 90°, the velocity in the y direction begins to decrease, as illustrated in the second figure of Figure 26. This occurs around 4.5 seconds simulation time. This explains the divergence of the nonlinear response in v to the linear model response. This indicates that the linear model may be valid well through a disturbance of 1 m/s from the hover state. The input to output transfer functions for the principal dynamics are listed in Appendix H.

C. LINEARIZATION OF THE HEAVE DYNAMICS

Attempts to linearize the heave dynamics using the same techniques outlined in section B did not provide accurate linear models. While the lateral and longitudinal dynamics are governed by the main rotor flapping, the heave dynamics are governed by the nonlinear thrust response to collective commands. The linearization approach that was used to model the heave response is system identification. The goal is to obtain a system identification model by providing the "n4sid" function input and output data. It should also be mentioned here that the heave dynamics in the climb and descent are quite different, as explained in the modeling sections earlier. Therefore, two heave models were obtained, one for climb and one for descent.

Figure 27 illustrates the climbing heave step response for the linear and nonlinear model responses.

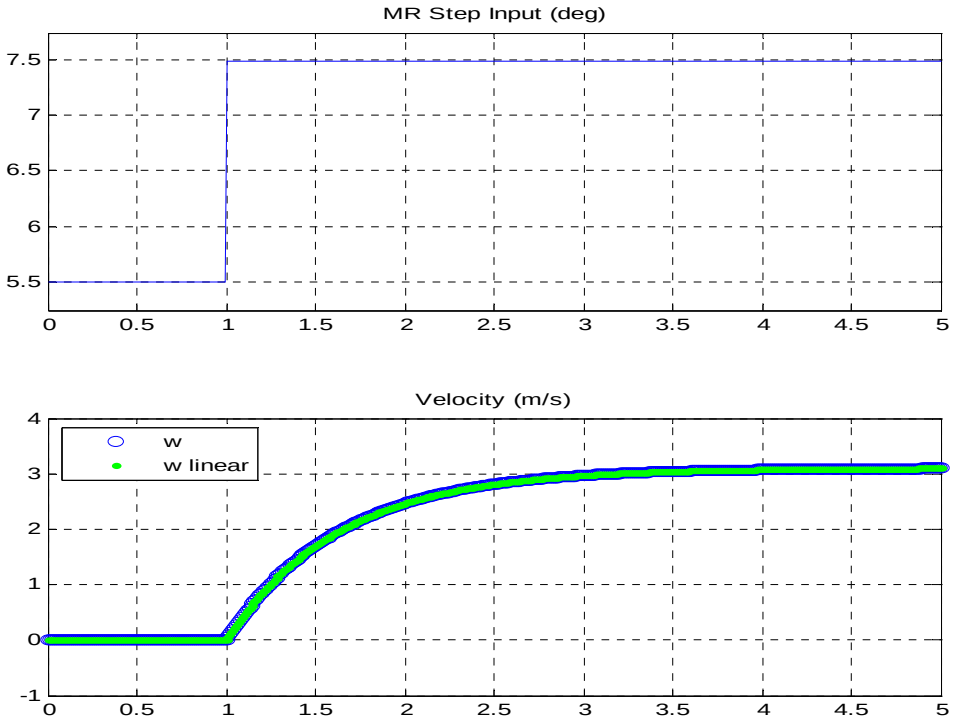


Figure 27. Heave Dynamics/Linear Climb Model Validation

Figure 27 illustrates the linear climbing heave model is accurate in the principal heave dynamics and the that the linear model is accurate in representing the nonlinear model up to 3 m/s climbing speed, given the 2° step collective input. In the flight path controller design, collective commands will only be sufficient to maintain the vehicle at a constant altitude, and therefore heave velocities will be well within the 3 m/s.

Figure 28 illustrates the linear descent heave model response compared to the nonlinear heave descent response.

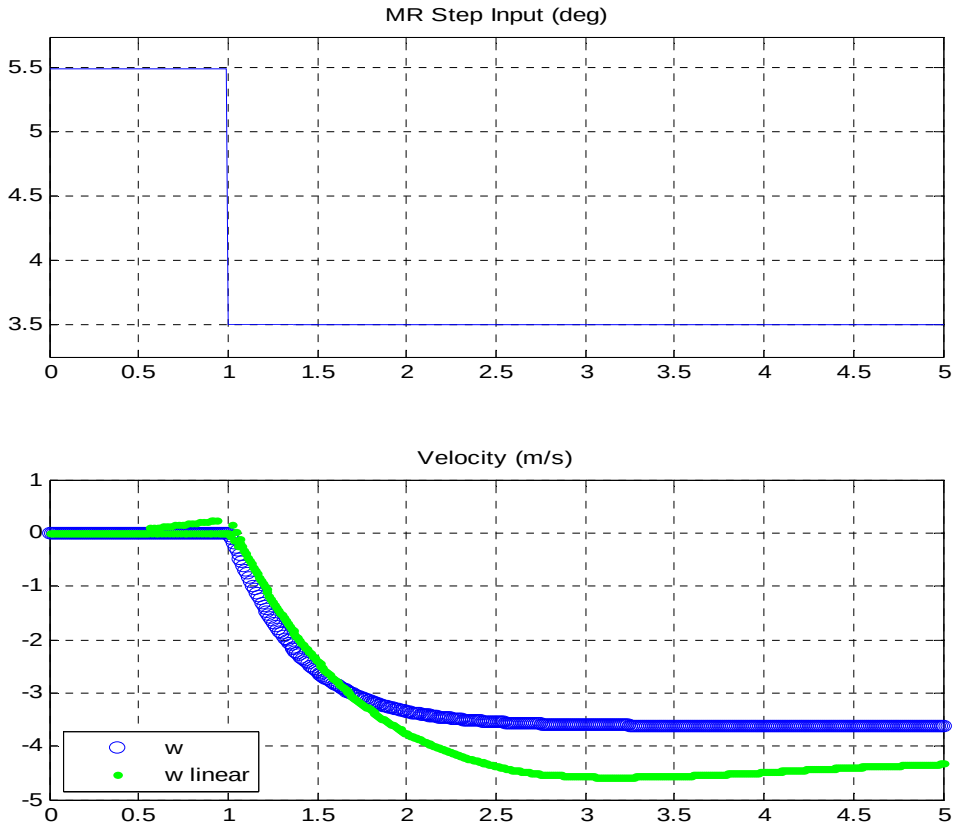


Figure 28. Heave Dynamics/Linear Descent Model Validation

Figure 28 illustrates that the linear heave descent model obtained diverges slightly from the nonlinear model at around 3 m/s descent. Inside of this boundary, the linear model represents the nonlinear dynamics very closely. The heave transfer functions for climb and descent are listed in Appendix H.

D. LINEARIZATION OF THE YAW DYNAMICS

The yaw dynamics are generated by the combination of the main rotor torque on the UAV body and the tail rotor thrust compensation. A stick command (left-right) for a tail rotor cyclic command will cause the tail rotor thrust to increase or decrease, depending on the type of command. The changing of the tail rotor thrust generates a force on the UAV tail, which causes the UAV to yaw. The linearization process that was applied for the lateral and longitudinal dynamics cannot be applied for the yaw degree of freedom. The approach that was taken here is system identification using a data driven modeling approach. The goal was to use simulation data from the nonlinear model to obtain a linear model of the yaw dynamics from the commanded tail rotor collective to the yaw angle ψ. Figure 29 illustrates the nonlinear yaw response and the linear model yaw response.

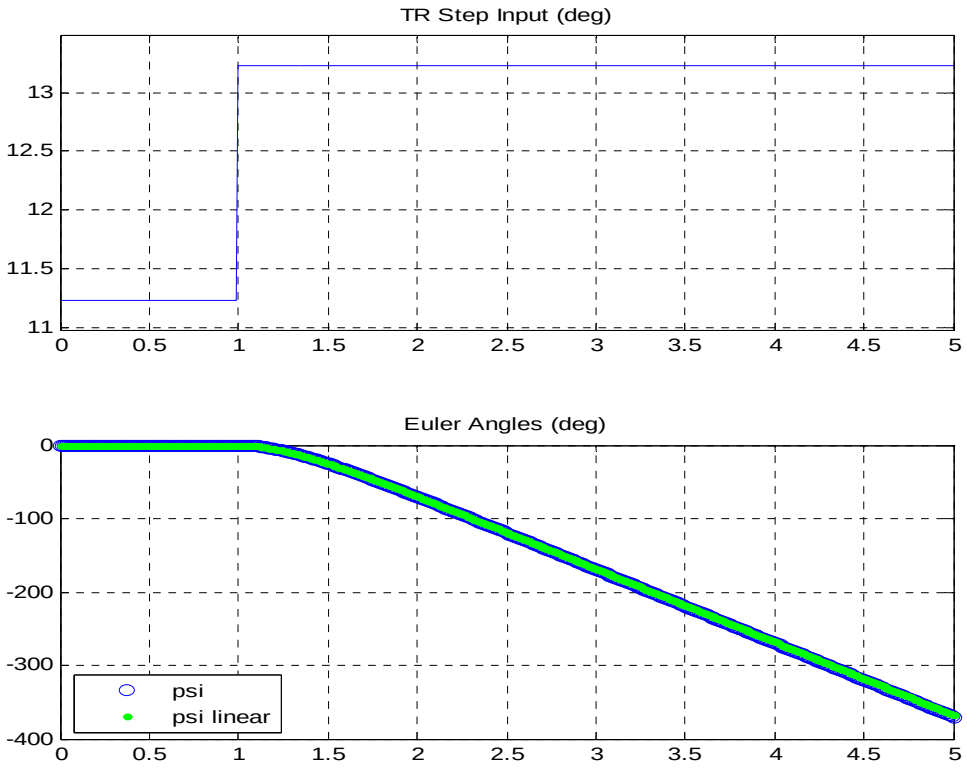


Figure 29. Yaw Dynamics Model Validation

Figure 29 illustrates that the linear yaw dynamics model obtained is accurate in the principal yaw dynamics, ψ. The yaw dynamics transfer function is listed in Appendix H.

In summary, the nonlinear model was linearized about the longitudinal and lateral modes using the Matlab function "linmod," while the heave and yaw modes were linearized using system identification; the system identification function "n4sid" facilitated this process. For a $2°$ flap angle step command, the linear model is accurate when compared to the nonlinear responses in the longitudinal and lateral modes. The step responses were accurate up to 1 m/s and they also indicate that linear models will provide accuracy well through 1 m/s. This was not verified and for this reason, flight path tracking performance of the PD gains to be derived will remain within this flight regime. The linear model for heave is also accurate up to 3 m/s, in both climb and descent. The linear model for the yaw mode is also very accurate for a $1°$ step command in tail rotor collective.

The next chapter will focus on the development of the closed loop system and tuning of the PD controllers for the longitudinal, lateral, heave, and yaw modes.

V. CONTROL DESIGN

A. CLOSED LOOP SYSTEM STRUCTURE

Figure 30 illustrates the nonlinear model with a lead Proportional Derivative (PD) compensator. The PD compensator receives the error signal, which is the difference between the reference signal (desired position) and unit feedback output (actual position).

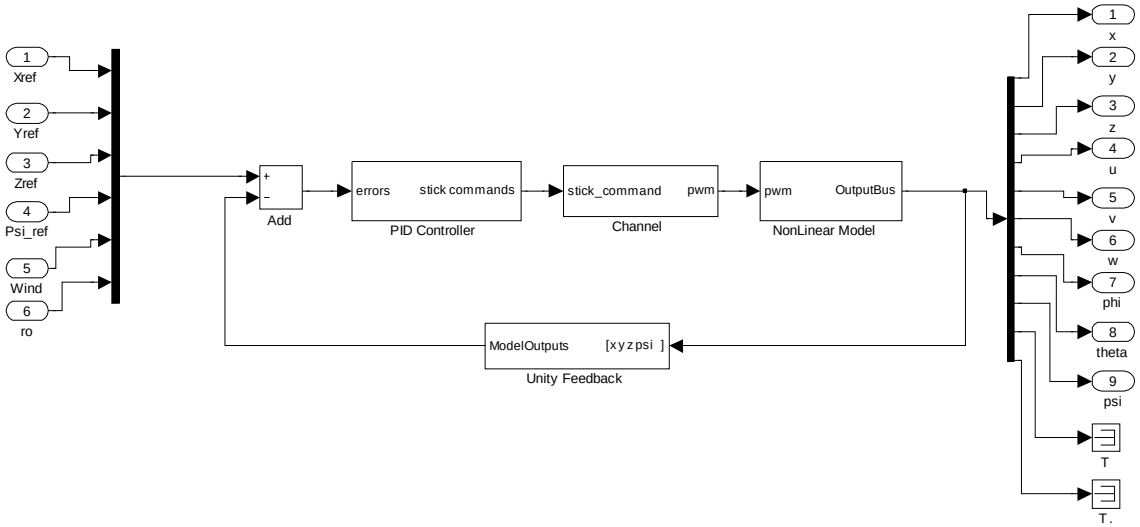


Figure 30. Closed-Loop System with Lead PD Compensation

The primary degrees of freedom to be controlled are longitudinal, lateral, heave, and yaw. This is achieved by implementing four Proportional Derivative (PD) controllers for the aforementioned modes.

B. DETERMINATION OF PD CONTROLLER GAINS

The Ziegler-Nichols tuning method outlined in [8, p. 673] was applied in order to obtain appropriate PD gains. The Ziegler-Nichols tuning method starts with all gains at zero and the test gain K_T is increased until sustained oscillations are obtained. A sustained oscillation is characterized by an oscillatory output that is constant in amplitude and period [8, p. 672]. Once the oscillation period P_c and test gain K_T at this point are known, Table 5 is applied in determining all of the initial estimate gains.

Control Type	K_P	K_I	K_D
P	$.5K_T$		
PI	$.45K_T$	$1.2K_P / P_c$	
PID	$.6K_T$	$2K_P / P_c$	$K_P P_c / 8$

Table 5. Ziegler-Nichols PID Gain Estimates (After [8])

To determine the initial longitudinal PD gains, a step input X_{ref} (input port 1) is provided to the closed loop system illustrated in Figure 30. With no longitudinal PD compensation $(K_P = K_D = 0)$, a reference step input in the longitudinal gives no output position signal x (output port 1), since the error signal into the channel is zero. The test gain K_T is then increased to $K_T = .1$ and the output position analyzed.

Figure 31 illustrates the position response to a reference position step input with $K_T = .1$ and oscillation period $P_C = 11$ sec. Figure 31 illustrates that a sustained oscillation has been obtained with $K_T = .1$.

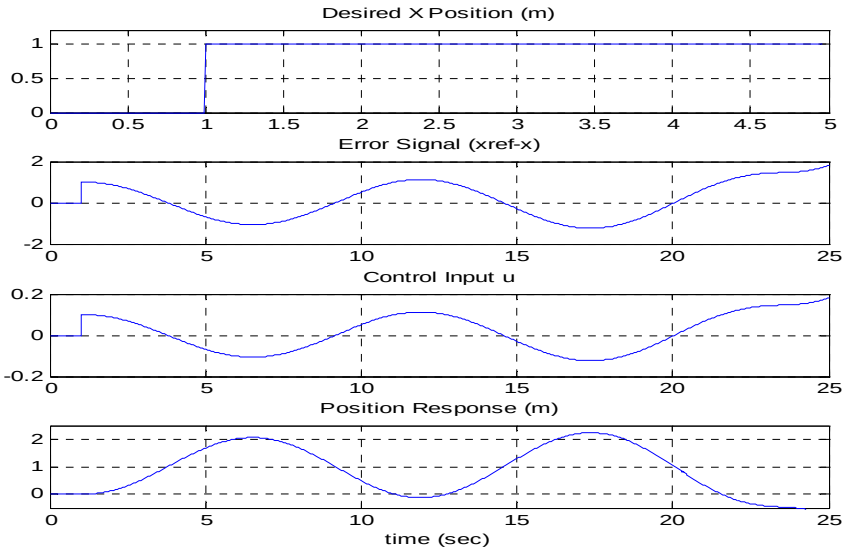


Figure 31. Position Response ($K_T = .1$)/ PD Tuning

Figure 31 illustrates the error signal along with the control input u that is generated from the proportional gain. This shows that the position response is oscillating about the desired response of 1 meter at the same frequency as the control input u. It also shows that the error signal, which drives the control input, is too large and is why insufficient control effort is being applied in this case.

Using the PID row of Table 5, without integral control, gives the initial longitudinal PD gain estimates of $K_P = .06$ and $K_D = .825$. An integral gain is not used since doing so made the closed loop response go unstable. Figure 32 illustrates the position response with the initial PD gain estimates.

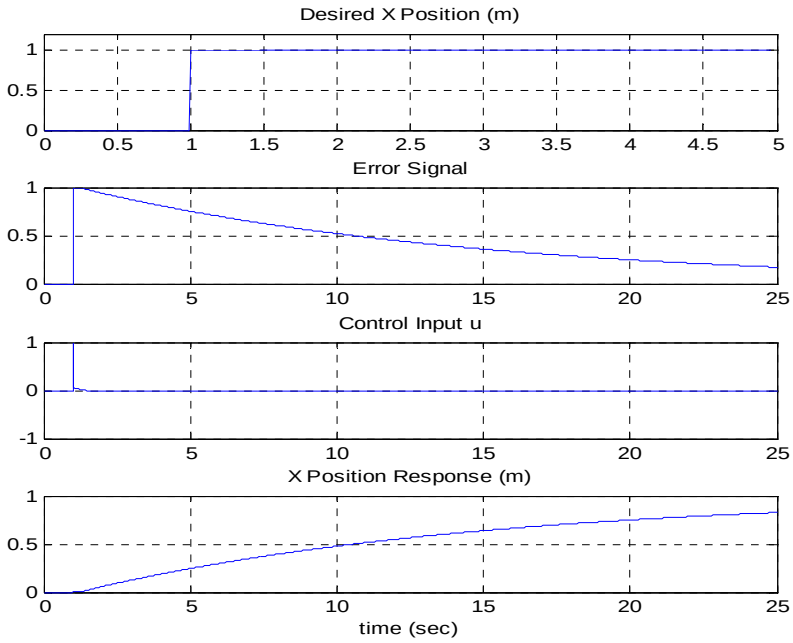


Figure 32. Position Response ($K_P = .06$ $K_D = .825$)/ Longitudinal Mode

Figure 32 shows that the error starts off as 1 meter and damps down exponentially towards zero. This is generated by an initial control impulse at 1 second with a peak of .8. The total time over which the control input is applied is roughly .35 seconds.

Figure 32 also shows that the response to a step input of 1 meter is too slow. The desired response should yield a settling time inside of 3 seconds with an overshoot of less

than 5%. Tuning the gains further gives $K_P = 3.0$ and $K_D = 1.6$ with the step responses illustrated in Figure 33. The lateral dynamics are very similar to the longitudinal and using the same gains provided similar results.

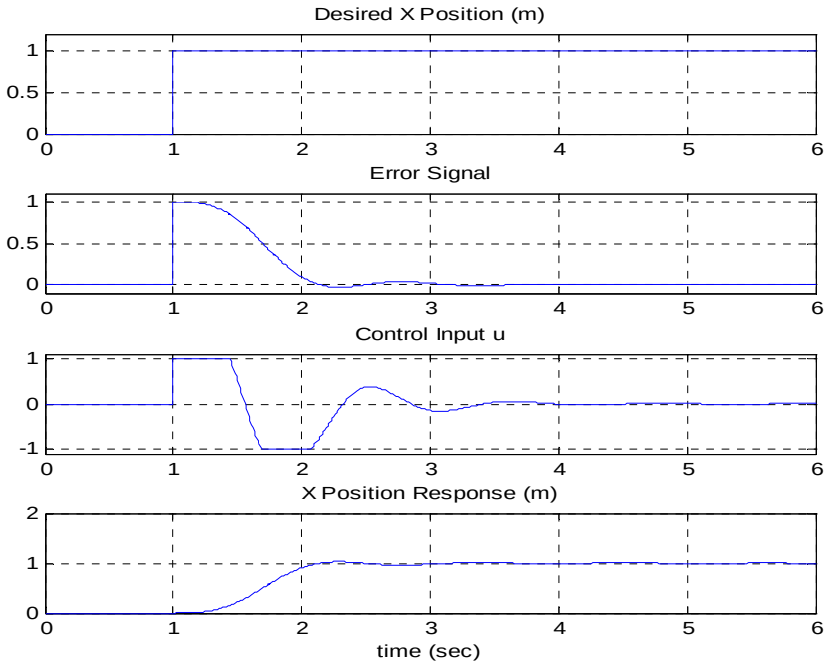


Figure 33. Position Response ($K_P = 3.0$ $K_D = 1.6$)/ Longitudinal Mode

Figure 33 illustrates that the error between the desired and actual position damps down to within 2% of zero error 1.11 seconds after the reference signal is commanded. The determined PD gains yield a settling time less than 3 seconds and an overshoot less than 3%. Figure 34 shows the velocity response along with the position.

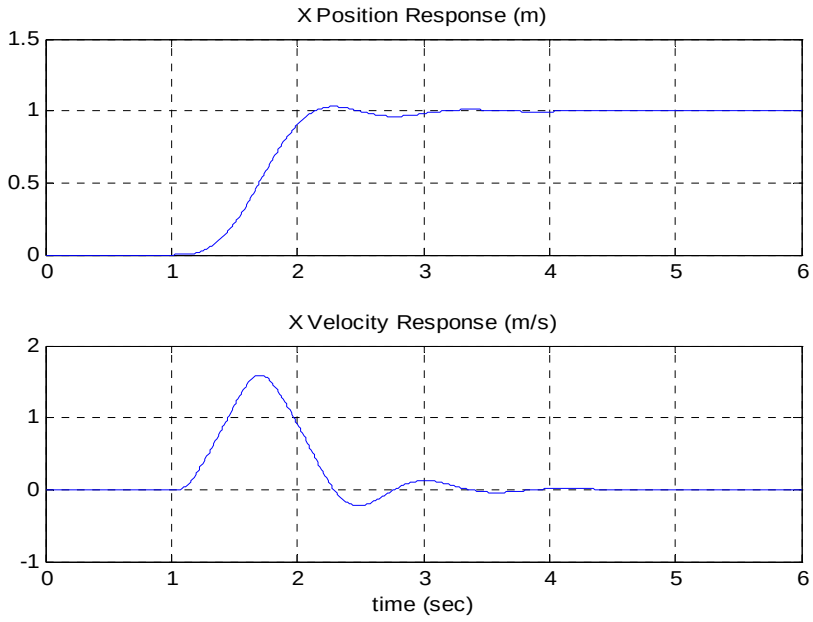


Figure 34. Velocity Response ($K_P = 3.0$ $K_D = 1.6$)/ Longitudinal Mode

The same tuning procedure is performed for the heave PD controller gains. Figure 35 illustrates the Z velocity response along with the PD compensator error signal and resultant control signal u .

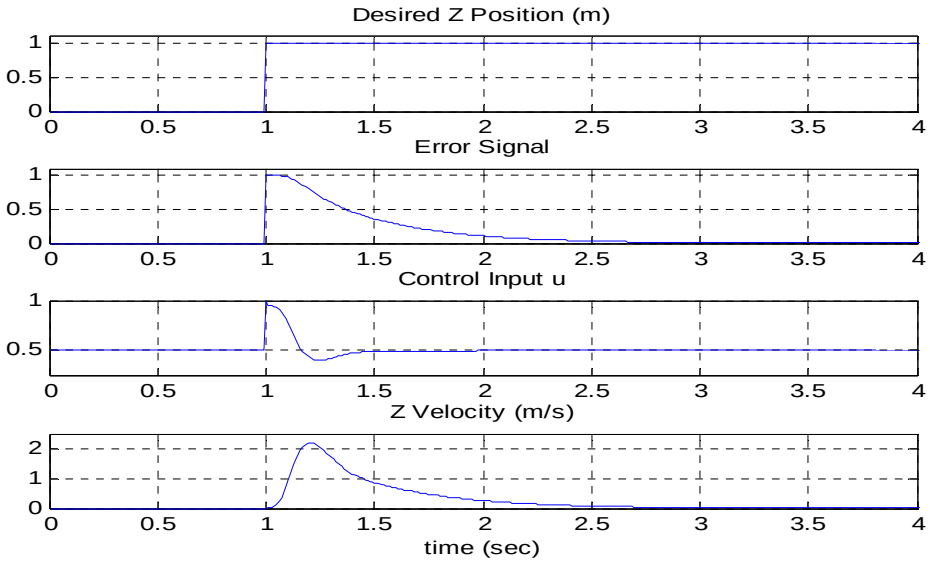


Figure 35. Position Response ($K_P = .45$ $K_D = .2$)/ Heave Mode

The heave response gives a settling time less than 2 seconds with no overshoot.

The yaw mode controller gains yield the responses illustrated in Figure 36.

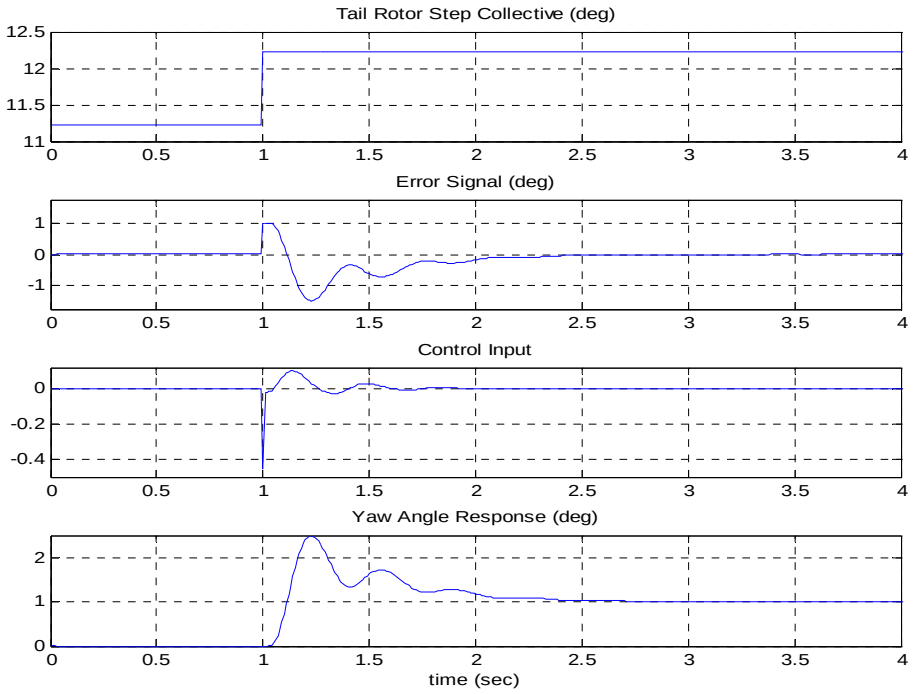


Figure 36. Yaw Angle Response ($K_P = .1 \;\; K_D = .05$)/ Yaw Mode

The yaw angle response gives excessive overshoot but this can be compensated for in the future by including a yaw rate feedback. For now, this will be sufficient to provide a steady orientation. Table 6 outlines the PD gains for all four modes.

MODE	K_P	K_D
Longitudinal	3.0	1.6
Lateral	3.0	1.6
Heave	.45	.2
Yaw	1.1	.25

Table 6. PD Gains at Hover

C. FLIGHT PATH TRACKING PERFORMANCE / SQUARE PATH

The PD controller gains listed in Table 6 are tested on the nonlinear model by feeding the closed loop system a table of reference inputs that represent position commands in the three dimensions. The following will analyze the controller tracking performance.

1. Position Tracking Performance

The flight path trajectory that was set up for this test is a 4m×4m square at a 2-meter altitude. The initial position of the model is set at the origin and the first leg of the square path is along the y-axis. Figure 37 illustrates the x and y position response of the model versus the reference path.

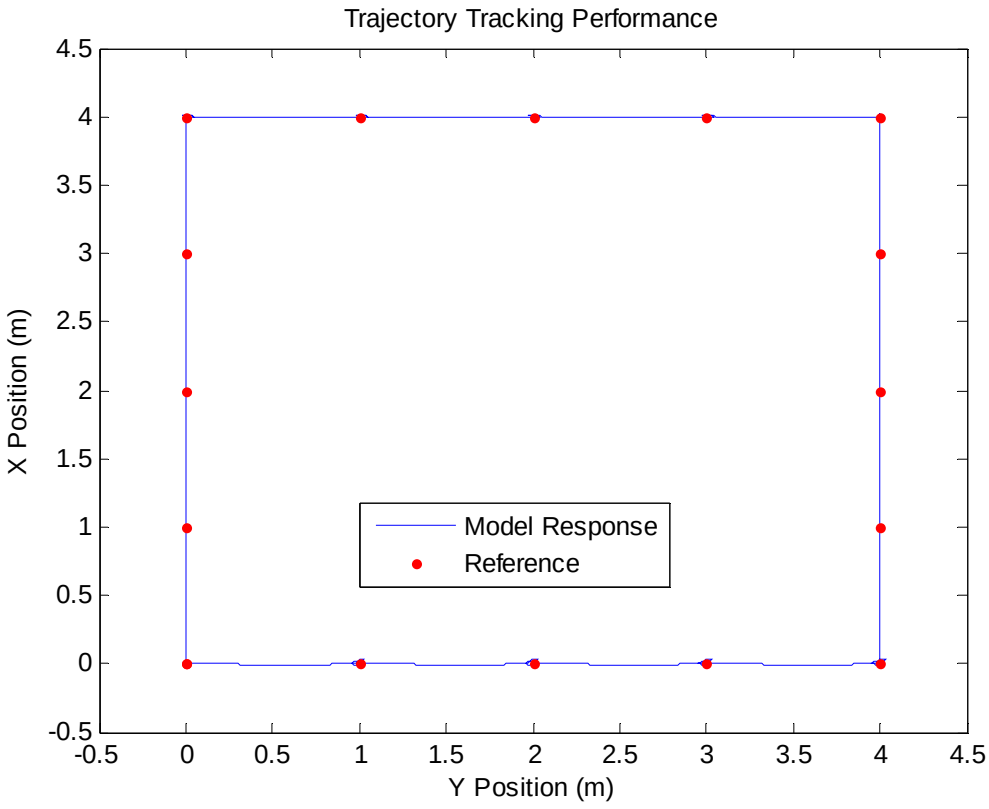


Figure 37. Flight Path Tracking Performance/Square Path

Figure 37 illustrates that the longitudinal and lateral mode PD controller performs well. The maximum position error observed is 3.1 cm and is more easily identified in Figure 38, which is a zoomed in version of Figure 37, for the first leg of the track.

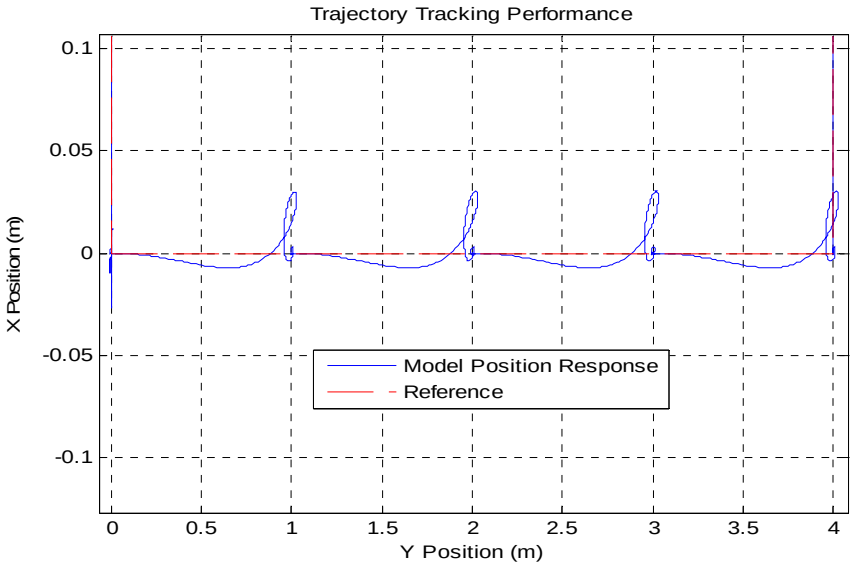


Figure 38. Flight Path Tracking Performance/Error

Figure 39 illustrates the nonlinear model's flight path in three dimensions. Figure 40 illustrates the flight path trajectory with increased vertical position resolution, which shows that the maximum vertical position error is about 1.4 cm.

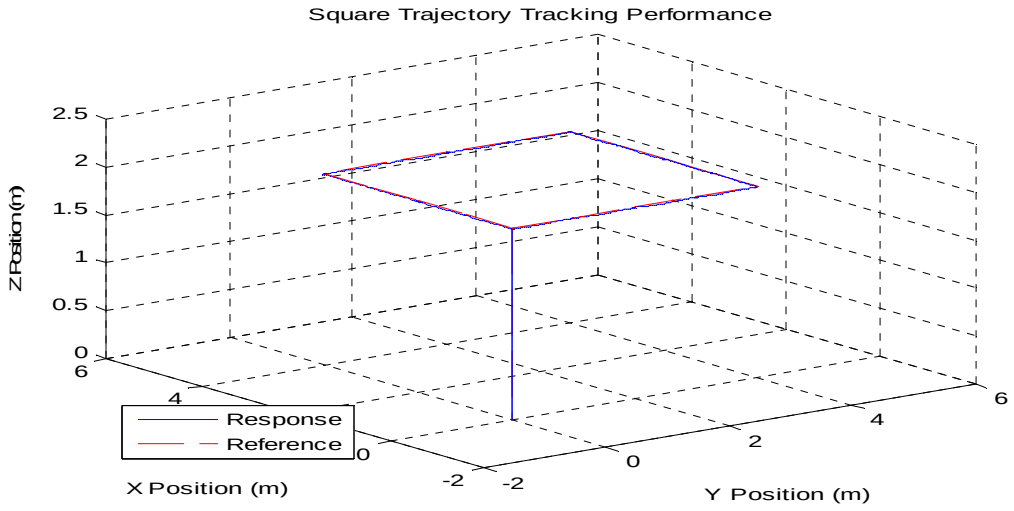


Figure 39. 3D Flight Path Tracking Performance/Square Path

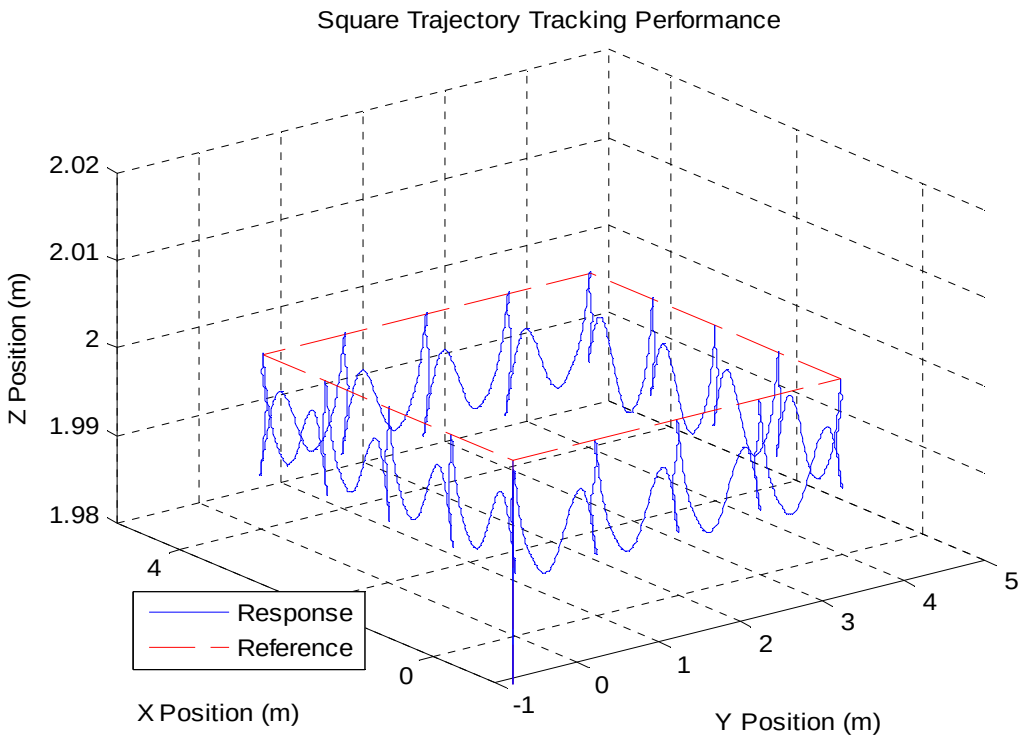


Figure 40. 3D Flight Path Tracking Performance/Vertical Error

2. Yaw Control Performance

Figure 41 illustrates the yaw response (in degrees) for a commanded yaw angle of zero degrees. Initially, as the helicopter is climbing to 2 meters, the tail boom is rotating clockwise because of the increasing main rotor moment on the body and explains the increasing yaw angle for the first three to four seconds. The remaining 25 seconds of the response illustrated in Figure 41 represents the first leg where the helicopter model is traveling from left to right, along the inertial frame's positive y-axis. The reference positions are commanded in a way that causes the helicopter to surge in 1 meter increments. Then, the helicopter slows down as it approaches its final position, and hovers until the next 1 meter command is given. This explains the 4 yaw angle spikes illustrated in Figure 41. As the model travels in its sideways trajectory (1 meter at a time), the tail boom has the tendency to rotate clockwise and explains the increasing yaw angle. As the yaw angle increases, the PD controller applies control effort and brings the yaw angle back to zero, before the model begins another 1 meter surge in the positive y

direction. The yaw angle spikes can be reduced, by applying a yaw rate feedback to the nonlinear model. Figure 42 illustrates the yaw angle responses for the entire flight path tracking simulation and shows that the yaw angle errors are smaller, for the second and fourth legs of the flight path, because of the flight path orientation with respect to the helicopter tail boom.

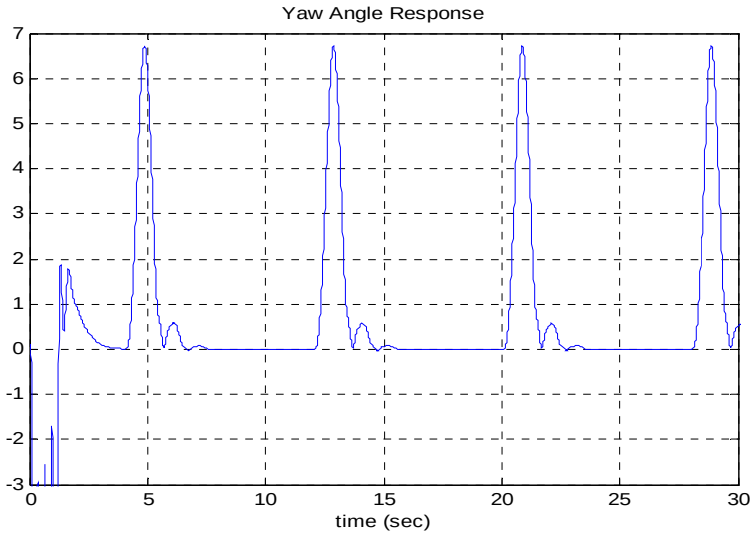


Figure 41. Yaw Angle Response/First Leg

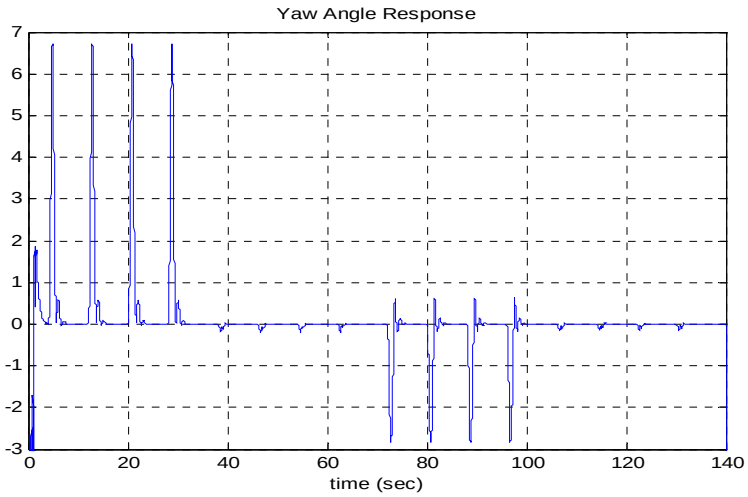


Figure 42. Yaw Angle Response/Entire Path

3. Euler Angle Responses

This model does not implement a direct control scheme for the UAV orientation; rather, it relies on the inherent stability that has been modeled. The UAV has been modeled so that an angle disturbance will be compensated for by the hub spring force as much as possible without having undesired effects. Figure 43 illustrates the model position response in the y direction along with the roll angle response, for the first leg of the simulation.

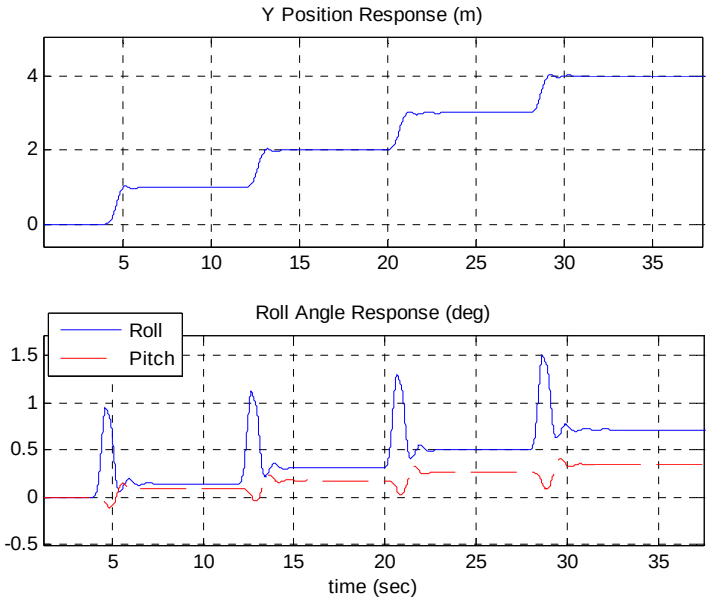


Figure 43. Roll Angle Response/First Leg

Figure 43 illustrates that with each successive surge in the y direction, the roll angle increases in steps. With each surge, the roll angle increases then damps back to a smaller roll angle that is larger than the initial roll angle. Also shown is the pitch angle which is only slightly increasing (positive pitch means nose up) due to rotor flap angle coupling. Figure 44 illustrates the y position response and the roll angle responses for the entire simulation. It is evident that the roll angle begins to decrease back towards zero as the UAV travels in the –y direction.

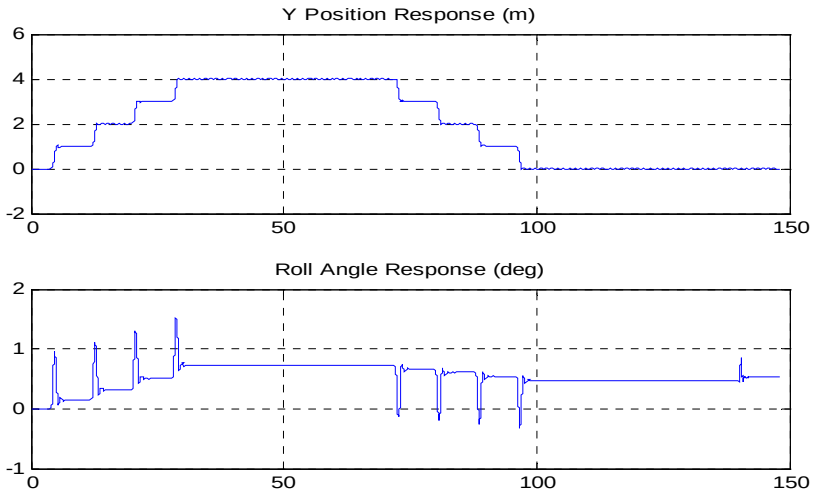


Figure 44. Roll Angle Response/Entire Simulation

Figure 45 shows similar results for the pitch angle responses. As the x position increases, the pitch angle becomes more negative (nose down). The pitch angle begins to recover as the UAV flies backward along the x-axis.

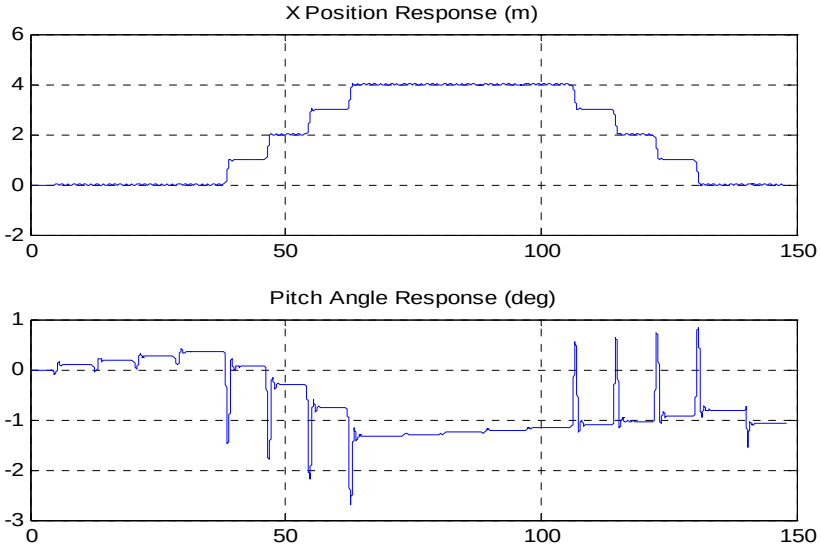


Figure 45. Pitch Angle Response/Entire Simulation

D. FLIGHT PATH TRACKING PERFORMANCE / FIGURE 9

A figure-8 reference flight path is generated with user specified radius, number of rotations for each loop, and desired flight time for each complete circle. Initially, the model begins at ground level and climbs to 5 meters and once positioned at [x y z]=[2 0 5], the figure-8 path tracking routine begins. The radius chosen for this demonstration is 1 meter with the initial position of [2 0 5]. The desired time to traverse the 360° for each loop is defined as 10 seconds.

1. Position Tracking Performance

Figure 46 illustrates the top view of the helicopter figure-8 trajectory versus the commanded reference trajectory.

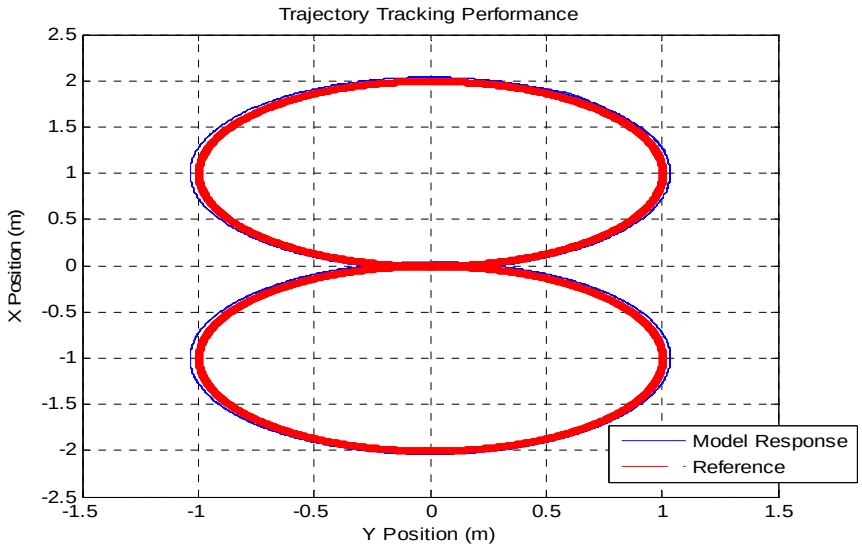


Figure 46. Flight Path Tracking Performance/Figure 9 (Top View)

Using the same PD gains derived earlier gives acceptable tracking performance with a maximum position error of 3.6 cm. Figure 47 illustrates the three dimensional flight trajectory with the initial climb command.

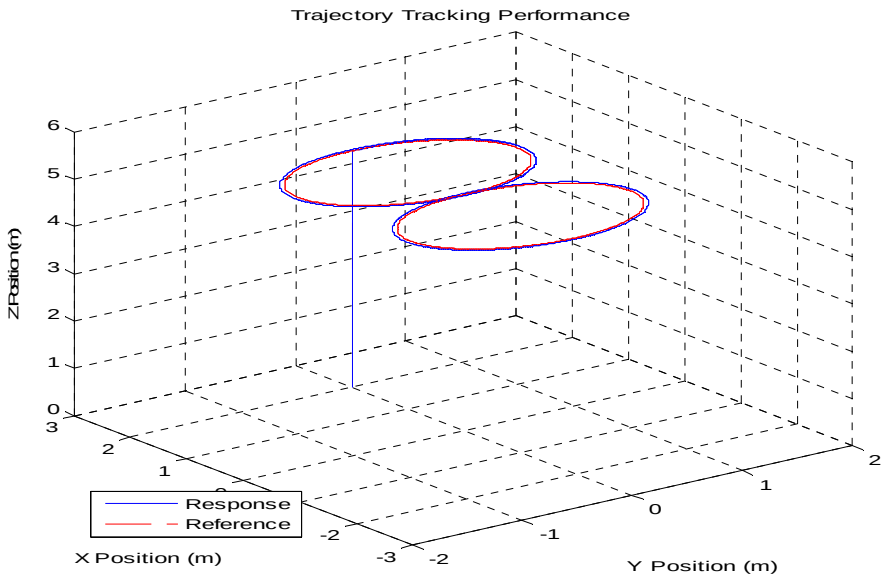


Figure 47. Flight Path Tracking Performance/Figure 9 (3D)

Figure 48 is a zoomed-in version to show more resolution in the vertical axis. The maximum error in the vertical position is about 1.7 mm after an initial maximum error of 8.6 mm.

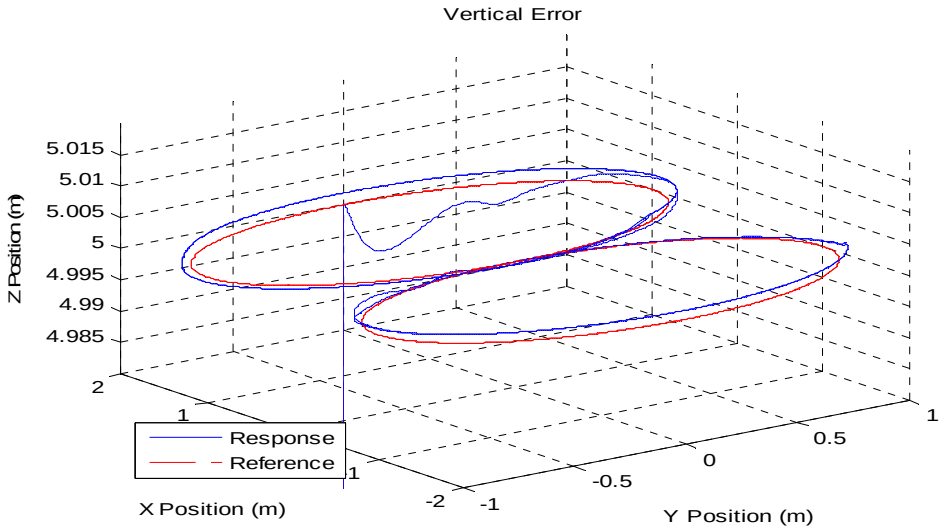


Figure 48. Flight Path Tracking Performance/Figure 9 (Vertical Position Error)

Figure 49 illustrates the velocity responses u and v.

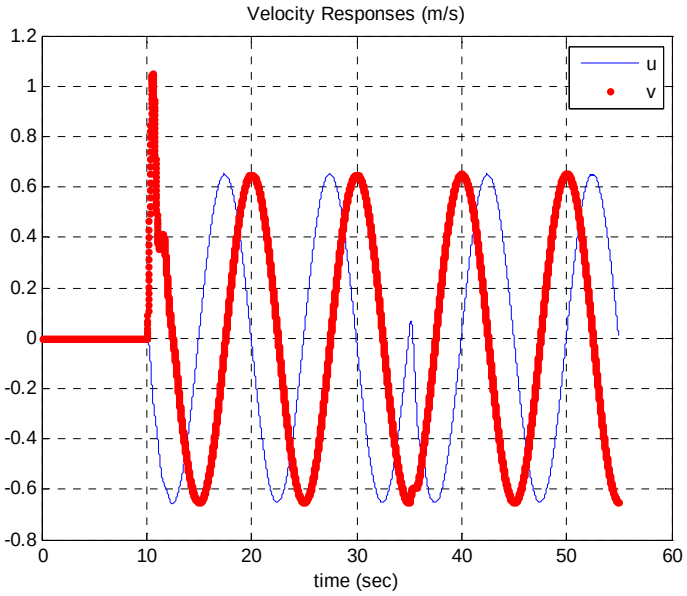


Figure 49. Flight Path Tracking Performance/Figure 9 (Velocity Responses)

2. Yaw Control Performance

Figure 50 illustrates the yaw angle response to a commanded angle of zero degrees. The swaying of the tail boom is natural in that the tail has the tendency to rotate anytime the vehicle turns in one direction or another. If the vehicle turns right, the tail will rotate clockwise and if the vehicle turns left, the tail will rotate counter clockwise. In this figure-8 trajectory, the vehicle is constantly turning and is why the tail is constantly oscillating. In this simulation, the tail does not sway more than $4.5°$ once established on the Figure-8 trajectory. As mentioned earlier, however, the tail boom yaw angle error can be reduced significantly by implementing a yaw rate feedback in future control projects.

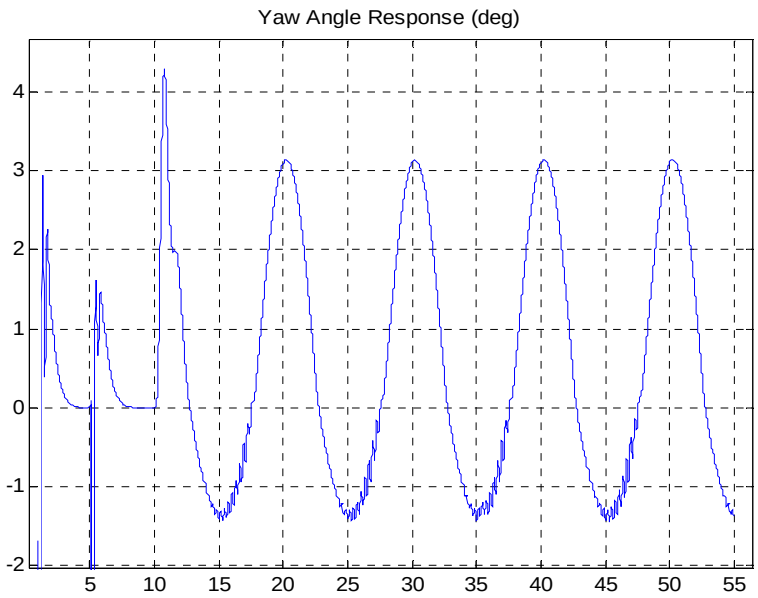


Figure 50. Yaw Angle Response/Figure 9

3. Euler Angle Responses

Figure 51 illustrates the roll angle response with respect to the y position.

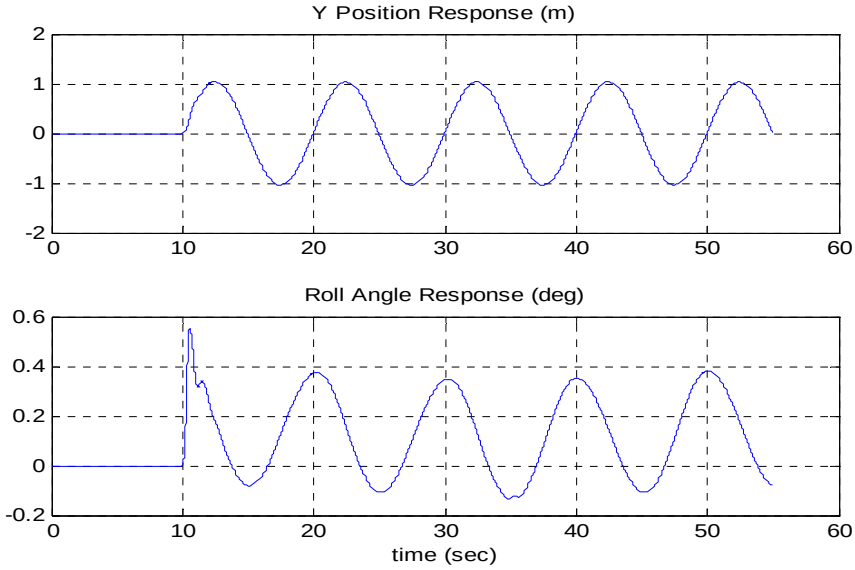


Figure 51. Roll Angle Response/Figure 9

From about 12 seconds to 15 seconds, Figure 51 illustrates the vehicle y position. In this time frame, the vehicle starts at $y = 1$ m and flies towards $y = 0$. Analyzing the bottom figure of Figure 51, along the same time frame, shows that the roll angle is decreasing (rolling towards the left side) and is consistent with the desired roll angle response. In traveling from $y = -1$ m back to $y = 0$, the roll angle is increasing (rolling to the right). These observations are consistent with the expected roll angle responses.

Figure 52 illustrates the pitch angle response along with the corresponding x position.

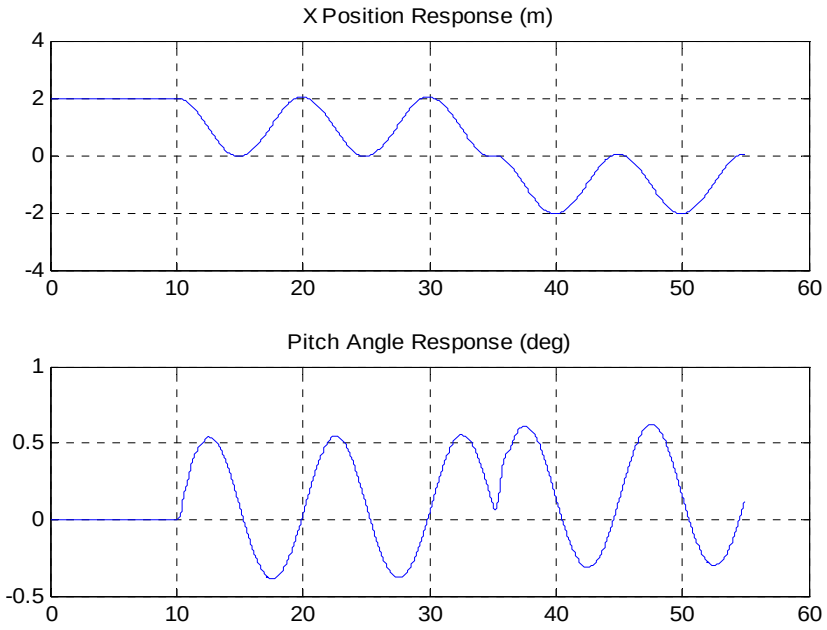


Figure 52. Pitch Angle Response/Figure 9

Figure 52 illustrates that as the vehicle position along the x axis decreases (backwards flight), the pitch angle increases (increasingly nose up attitude). Conversely, as the vehicle travels along the $+x-$ axis (forward flight), the nose pitches down for a decreasing pitch angle.

E. CONCLUSIONS ON TRACKING PERFORMANCE

The position tracking performance has given nice results with maximum errors of 3.6 cm for translational motion and 8 mm for vertical position. The yaw angle errors are expected given that no yaw rate feedback is modeled at this point. The actual plant does have a yaw gyro that provides rate feedback to the sensors. Mettler's work [2] is a good source for obtaining yaw rate feedback models. The angle responses are generally correct in that main rotor flap commands generate the roll and pitch responses that are expected. Flight in the $\pm x$ direction, for example, generates the anticipated pitch angle responses. Conversely, flight in the $\pm y$ direction generates the anticipated roll angle responses. Getting the specific quantitative RW UAV roll and pitch angle responses will require system identification of the MAE Department miniature helicopter so that the flapping spring derivatives M_a and L_b can be determined. As explained earlier, these derivatives determine the way that aerodynamic moments from the main rotor are cancelled by restoring moments generated by the blade centrifugal forces and by damping moments provided by the main rotor stabilizer bar.

VI. CONCLUSION

A. MODEL DISCREPANCIES

The current model can only perform for smaller commanded yaw angles, from $0°$ to about $20°$. Beyond $20°$, the position errors get larger as the model moves through its flight path and will even enter into spiraling motion and or divergence from the desired flight path. Further tests will have to be conducted to determine the cause of this unexpected behavior. The tail rotor model seems to be doing fine on its own given that single inputs of varying yaw angles, with yaw controller, generate the expected yaw response. A step in yaw angle for $180°$ and $270°$ both gave over damped yaw angle responses with roughly 1-second settling time. Figure 53 illustrates the yaw angle response to a $270°$ step command.

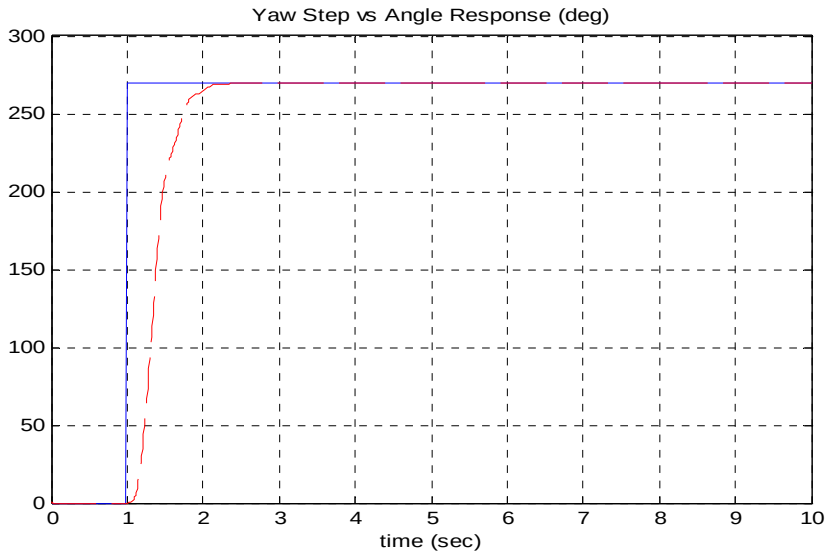


Figure 53. Yaw Step Response $\psi = 270°$

Figure 54 illustrates the control input generated by the PD controller in order to get the UAV to the $270°$ orientation.

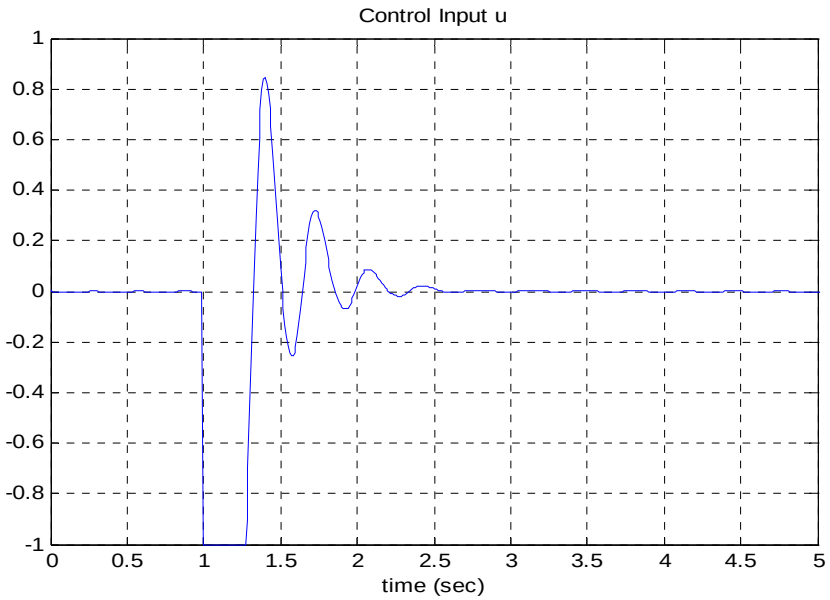


Figure 54. Yaw Step Command Control Effort

Figure 54 shows that the control effort is only required while getting the UAV to the desired orientation, from 1 to roughly 2.5 seconds, after the yaw angle has settled. The significance of the control signal illustrated in Figure 54 is that it shows that no control effort is required once the desired orientation is obtained. This is also in keeping with the fact that, once the desired orientation is obtained, and control effort taken out, the tail rotor thrust should return to the hover value of 7.29 Newtons, as discussed in Chapter III.C.2, p. 42. Figure 55 illustrates the tail rotor thrust response to the 270° step command.

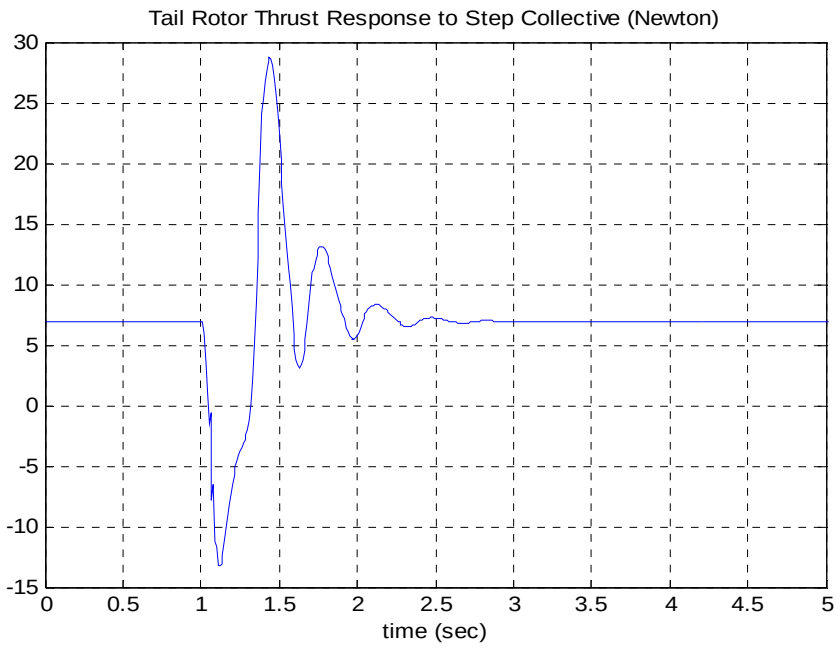


Figure 55. Tail Rotor Thrust Response to Yaw Step Command/Static

At 1 second, the tail rotor thrust is driven to almost -15 Newton in order to generate the necessary yaw moment required to make the UAV spin to $+270°$. As the vehicle approaches the desired yaw angle, counter-thrust is applied to keep the tail from overshooting; the thrust continues to oscillate until the tail rotor thrust converges back to the hovering tail rotor thrust of 7.29 Newton. Figures 52 through 54 support the validity of the tail rotor model design.

The problem arises when the yaw angle command is followed up with a position step command in either direction. Figure 56 illustrates the position response to the $X_{ref} = 1$ m position command. The position step command is given 4 seconds after the yaw step command of $270°$. This is done to ensure that sufficient settling time has been given to the yaw angle.

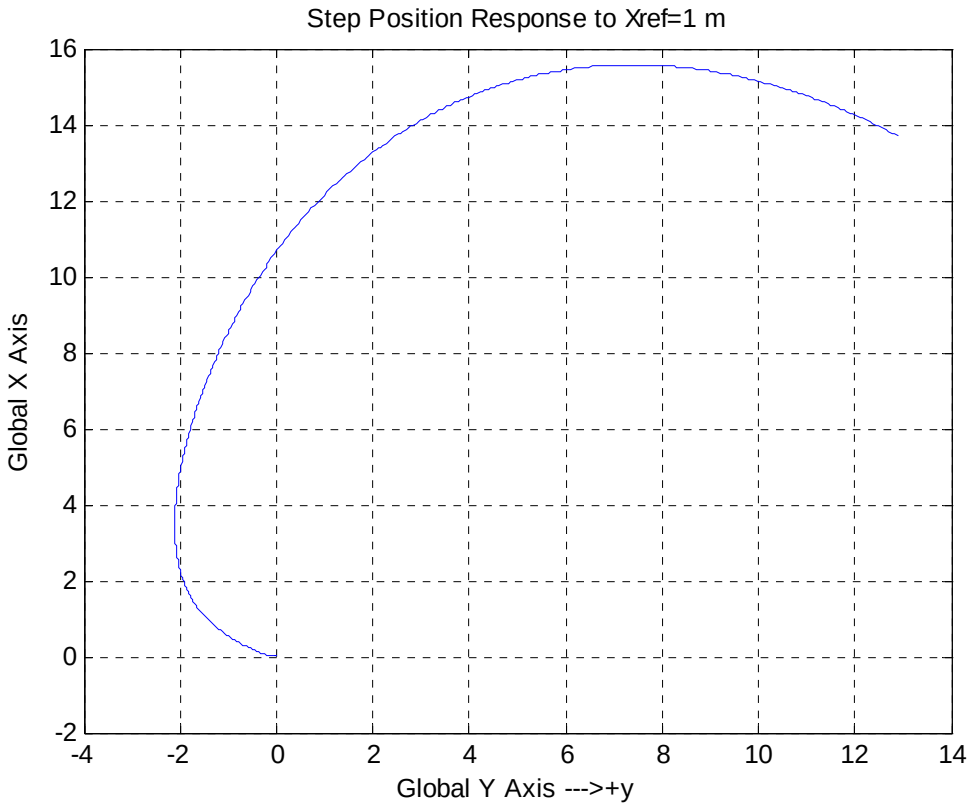


Figure 56. Position Response at $\psi = 270°$

By the time the model reaches the $270°$ orientation, the coordinate system for the model should be oriented such that the inertial frame x-axis is pointing in the direction of the earth frame's $-y$-axis. After the $270°$ turn, the step command in the x direction should result in the model flying towards the body fixed x direction or, the earth coordinate –y direction. What Figure 56 illustrates is that the model is going in the direction of the +x earth coordinate, before turning back into the –x earth coordinate direction.

The model discrepancy is unlikely to be due to tail rotor modeling. Instead, the most likely source of the error is in the inertial frame direction cosine matrix signals coming from the 6-DOF blockset. These signals will have to be analyzed, and signal gains may have to be applied in order to obtain the expected results.

All of the simulations using the box and Figure 9 are such that the earth coordinates and inertial frames are in alignment and is why the simulations from Chapter V worked fine for $\psi = 0°$.

B. FUTURE WORK

Future work would need to address the discrepancies between the body fixed frame and the earth coordinates, assuming that this is the source of simulation errors for large yaw angle commands.

Additional future work includes quantitative model verification from flight test data and iterative model tuning and configuration. The tuned model will need to be tested in other flight configurations in order to obtain state transition matrix derivative functions and robust control testing will need to be applied. Once the model has been fully validated and a robust controller satisfactorily tested, the next step will be to embed the control technique to the MicroPilot Autopilot for Hardware in the Loop testing.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A: MIT INSTRUMENTED X-CELL 60 SYSTEM IDENTIFICATION UAV PARAMETERS (AFTER [1])

m=8.2 kg	helicopter mass
I_{xx}=.18 kg m^2	rolling moment of inertia
I_{yy}=.34 kg m^2	pitching moment of inertia
I_{zz}=.28 kg m^2	yawing moment of inertia
Ω =167 rad/sec	Main rotor speed
$a_{mr} = 5.5$ rad^{-1}	Main rotor blade lift curve slope
C_{Do}^{mr} =.024	Main rotor zero lift drag coefficient
$h_{mr} = .235$ m	Main rotor hub height above c.g.
$l_{tr} = .91\ m$	Tail rotor hub location behind cg
$h_{tr} = .08\ m$	Tail rotor height above cg
$L_b = 166$ Nm/rad	Longitudinal flapping spring derivative at hover
$M_a = 82.6$ Nm/rad	Longitudinal flapping spring derivative at hover

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B: NEWTON-RAPHSON TECHNIQUE FOR ROTOR INFLOW ESTIMATION (M FILES)

getvi_v2.m

```
%getvi_v2.m
%Determine Main Rotor Inflow using Newton-Rhapson Technique
%Input: in=[u v w uwind vwind wwind T ro alpha vi[n+1]]
%Output: vi
function vi=getvi_v2(in)
i=1;
u=in(1);
v=in(2);
w=in(3);
uwind=in(4);
vwind=in(5);
wwind=in(6);
T=in(7);
ro=in(8);
alpha=abs(in(9));
vi_previous=in(10);
c=ro*((167*.775)^2)*pi*(.775^2);
CT=T/c;
vt=sqrt((u+uwind)^2+(v+vwind)^2);
mu=vt/(167*.775);
vz=w+wwind;
vh=4.171;
lh=4.171/(167*.775);
mud=vz/(167*.775);
if vz<0 && vz>-.834
    lamda(1)=lh*(1-mud/lh);
elseif vz<=-.834 && vz>-2*4.171
    lamda(1)=(.11108*vz+5.0976)/(167*.775);
elseif vz<=-2*4.171
    vi=-4.171*((vz/(2*vh))+sqrt(((vz/(2*vh))^2)-1));
    lamda(1)=vi/(167*.775);
elseif vz>=0
    lamda(1)=4.171/(167*.775);
    e=1;
    i=1;
    if i<500
        while abs(e)>=.00005
        flamda(i)=lamda(i)-mu*tan(alpha)-CT/(2*sqrt(mu^2+lamda(i)^2));
        flamda_prime(i)=1+CT/2*((mu^2+lamda(i)^2)^(-3/2))*lamda(i);
        lamda(i+1)=lamda(i)-flamda(i)/flamda_prime(i);
        e=abs((lamda(i+1)-lamda(i))/(lamda(i+1)));
        i=i+1;
        end
    elseif i>=500
    lamda(i)=vi_previous/(167*.775);
    end
end
vi=lamda(i)*167*.775;
```

getvi_v2TR.m

```
%getvi_v2TR.m
%Determine Tail Rotor Inflow using Newton-Rhapson Technique
%Input: in=[u v w uwind vwind wwind Ttr ro alpha=0 vi[n+1]]
%Output: vi
function vi=getvi_v2(in)
i=1;
u=in(1);
v=in(2);
w=in(3);
uwind=in(4);
vwind=in(5);
wwind=in(6);
T=in(7);
ro=in(8);
alpha=abs(in(9));
c=ro*((778.2*.13)^2)*pi*(.13^2);
CT=T/c;
vt=sqrt((u+uwind)^2+(v+vwind)^2);
mu=vt/(778.2*.13);
vz=w+wwind;
vh=7.294;
lh=7.294/(778.2*.13);
mud=vz/(778.2*.13);
if vz<0 && vz>-.834
    lamda(1)=lh*(1-mud/lh);
elseif vz<=-.834 && vz>-2*4.171
    lamda(1)=(.11108*vz+5.0976)/(778.2*.13);
elseif vz<=-2*4.171
    vi=-7.294*((vz/(2*vh))+sqrt(((vz/(2*vh))^2)-1));
    lamda(1)=vi/(778.2*.13);
elseif vz>=0
    lamda(1)=7.294/(778.2*.13);
    e=1;
    i=1;
    if i<500
        while e>.0005
        flamda(i)=lamda(1)-mu*tan(alpha)-CT/(2*sqrt(mu^2+lamda(i)^2));
        flamda_prime(i)=1+CT/2*((mu^2+lamda(i)^2)^(-3/2))*lamda(i);
        lamda(i+1)=lamda(i)-flamda(i)/flamda_prime(i);
        e=abs(lamda(i+1)-lamda(i))/(lamda(i+1));
        i=i+1;
        end
    elseif i>=500
    lamda(i)=7.294/(778.2*.13);
    end
end

vi=lamda(i)*778.2*.13;
```

UpdateMR.m

```
%UpdateMR
%          1  2  3   4     5     6   7  8   9     10     11   12   13
%Inputs:u=[u   v   w  uwind  vwind  wwind  ro  vt  cp  cp[n+1]  T[n+1]  vi
vi[n+1]]

function out=updateMR(in)
u=in(1);
v=in(2);
w=in(3);
uwind=in(4);
vwind=in(5);
wwind=in(6);

ua=u+uwind;
va=v+vwind;

ro=in(7);
vt=in(8);
cp=in(9);
cp_previous=in(10);
T_previous=in(11);
vi=in(12);
li=vi/(167*.775);
vi_previous=in(13);
li_previous=vi_previous/(167*.775);

fx=.1;
fy=.22;
Dx=fx*ro*ua*vt/2;
Dy=fy*ro*va*vt/2;
D=sqrt((Dx)^2+(Dy)^2);
alpha=(57.3*D*pi/180)/(80.414);

%Define Input to thrust function
%input_f_thrust=[u v w uwind vwind wwind vi cp ro cp[n+1] T[n+1]]
input_f_thrust=[u  v  w  uwind  vwind  wwind  vi  cp  ro  cp_previous
T_previous];
%output: Thrust
T=getThrustMR(input_f_thrust);

%Define Input to get induced velcity function
%Input: in=[u v w uwind vwind wwind T ro alpha vi[n+1]]
input_f_vi=[u v w uwind vwind wwind T ro alpha vi_previous];
vi=getvi_v2(input_f_vi);
out=[T vi];
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C: PARAMETER DEFINITION FILES

Config.m

```
%Fixed Parameters
mass=8.2;
g=9.80665;
Inertia=[.18 0 0;0 .34 0;0 0 .28] ;
ICParam.Ts=1/100; %Sample time period
SR=1/ICParam.Ts;  %Sample Rate
%================================
%Flapping Dynamics Constants
%================================
LN=3.92;                  %Lock Number
omega=167;                %MR speed rad/sec
KB=18.89729;              %hub torsional stiffness {NM/rad}
IB=.038;                  %mr blade flapping inertia {kg m^2}
tauf=.1;
%=======================================================
%Main Rotor Moment Derivatives
%=======================================================
Ma=82.6;
Lb=166;
%================================
%Vehicle Geometric Properties
%================================
hmr=.235;         %height of MR from cg position
Svf=.012;         %Vertical fin area
Rtr=.13;            %tail rotor radius
Ltr=.91;             %tail hub loc behind cg
htr=.08              %tr height above cg
%=====================================================
%Transmitter Duty Cycle Limits
%[.6E-3 1.52E-3 2.4E-3] sec duty cycle corresponds to
%[-1 0 1] input in stick
%=====================================================
duty_cycle_limits=[.6E-3 1.52E-3 2.4E-3];
lat_stick_positions=[-1 0 1];
lon_stick_positions=[-1 0 1];
pedal_positions=[-1 0 1];
collective_stick_positions=[0 .5 1.0];
duty_cycle_col=[.6E-3 .9E-3 2.4E-3];
servo_output=[-45 0 45];
MR_collective_range=[-3.88*pi/180 5.495*pi/180 15*pi/180];
TR_collective_range=[-20.0*pi/180 11.2291*pi/180 25*pi/180];
max_cyclic_command=20;
% Name of the MAT-file that will be generated
cfgmatfile = 'TestPlantcfg';
% Save workspace variables to MAT file
save(cfgmatfile);
% Output a message to the screen
fprintf(strcat('\n Aircraft configuration saved as:\t',
strcat(cfgmatfile),'.mat'));
```

```
fprintf('\n');
```

CreateModelStructure.m

```
%CreateModelStructure.m
%==========================================
%Default Initial Conditions: Hovering
%==========================================
ICParam.pos=[0 0 0];
ICParam.vel=[0 0 0];
ICParam.euler=[0 0 0];
ICParam.pqr=[0 0 0];
ICParam.flap_angle=[0 0];
ICParam.MRThrust=80.414;
ICParam.MRInflow=4.171;
ICParam.MRCollectivePitch=5.495*pi/180;
ICParam.TRInflow=7.294;
ICParam.TRThrust=6.92;
ICParam.YawMoment=0.0;
ICParam.ro=1.225;
ICParam.Wind=[0 0 0];

%Simulink Model to Trim
%ICParam.SimModel=input('Enter name of model to run and trim:')
ICParam.SimModel='NonLinearTRexAlign';
fprintf('\nThe model order is being determined...\n');

%Get the sim options structure
%To run simget, ensure the model is set to fixed step in the
%Simulation>>Configuration Parameters dialogue, with the step time set
%to ICParam.Ts defined above
runtime=input('Enter the Simulation Run time (sec)\n');
ICParam.SimOptions=simget(ICParam.SimModel);
LatCyc=0;
LonCyc=0;
TRCol=11.2291;
MRCol=5.495;
TrimInput=[LonCyc LatCyc MRCol TRCol]*pi/180;

%Run Model for 1 sample period
fprintf('Please wait, running simulation at hovering conditions...')
[SimTime,SimStates,SimOutputs]=sim(ICParam.SimModel,[0 .01],...
       ICParam.SimOptions,[0 TrimInput;.01 TrimInput]);

%Find the state order
InitialStates=[ICParam.pos';ICParam.euler';ICParam.vel';ICParam.pqr';..
.
       ICParam.flap_angle'];
ICParam.NHeliStates=length(InitialStates);
ICParam.NSimulinkStates=length(SimStates(1,:));
ICParam.StateIdx=zeros(ICParam.NHeliStates,1);
clear SimTime SimStates SimOutputs
fprintf('Done.\n');

clear SimTime SimStates SimOutputs;
```

APPENDIX D: TRIMMING PROCEDURE FILE

TrimHeli.m

```
%TrimHeli.m
%%%Determine The Initial Guess For Aircraft Controls
%Choose Flight Condition to trim
fprintf('Enter 1 to trim at Hover, 2 to trim at other specific
states\n')
condition=input('Enter your choice:\n');

if condition==1
    ICParam.vel=[0 0 0];
    ICParam.pos=[0 0 0];
    ICParam.euler=[0 0 0];
    ICParam.Wind=[0 0 0];
elseif condition==2
fprintf('\n');
fprintf('\nChoose Steady State Flight Condition to Trim:');
fprintf('\n-----------------------------------------------\n')
ICParam.vel=input('Enter Desired Velocity States [u v w]:');
ICParam.pos=input('Enter Desired Position [x y z]:');
ICParam.euler=input('Enter Desired Euler Angles [fi theta psi] in
deg:')*pi/180;
ICParam.Wind=input('Enter the Wind Vector to apply to the model [uw vw
ww]:');
%Note, pqr and flap angles must be zero in steady state
end

%Trim Error Threshold
MaxErrVel=[.05 .05 .05];  %Not more than 5cm/sec in error
MaxErrEuler=[1 1 1]*pi/180; %Not more than .25deg in angle error
%MaxErrVel=[.2 .2 .2];
%MaxErrEuler=[3 3 3]*pi/180;
%Control Surface Gains
vel=max(ICParam.vel);
K=-.5/15*vel+1;
GainLat=-.01;
GainLon=.01;
GainTR=.01;
GainCollective=-.01;

fprintf('\n Computing the initial estimates for the required trim
inputs...\n');

GoodGuess=0; Niter=1;
Gain=[GainLon GainLat GainCollective GainTR];
goodguess=[0 0 0 0];
while (GoodGuess==0)&&(Niter<50)
%Velocity Loop
    %Run Simulink Model for a short time (10 sec)
    fprintf('Please wait, running simulation and comparing simulation
states with desired states...')
```

```
    [SimTime,SimStates,SimOutputs]=sim(ICParam.SimModel,[0 2],...
       ICParam.SimOptions,[0 TrimInput; 2 TrimInput]);
    ErrVel=SimOutputs(end,4:6)-ICParam.vel;
    ErrEuler=SimOutputs(end,7:9)-ICParam.euler;
    fprintf('\nIteration #%2d\n',Niter)
    fprintf('---------------\n')
    fprintf('\nX Velocity Error=%2.3f m/s\n',ErrVel(1))
    fprintf('Y   Velocity Error=%2.3f m/s\n',ErrVel(2))
    fprintf('Z   Velocity Error=%2.3f m/s\n',ErrVel(3))
    fprintf('\nRoll Angle Error=%2.3f (deg)\n',ErrEuler(1)*180/pi)
    fprintf(' Pitch Angle Error=%2.3f (deg)\n',ErrEuler(2)*180/pi)
    fprintf('   Yaw Angle Error=%2.3f (deg)\n',ErrEuler(3)*180/pi)

    for i=1:3;
        if abs(ErrVel(i))<=MaxErrVel(i)
            %TrimInput(i)=TrimInput(i);
            goodguess(i)=1;
        else
            TrimInput(i)=TrimInput(i)+ErrVel(i)*Gain(i);
        end
    end

    if abs(ErrEuler(3))<=MaxErrEuler(3)
        %TrimInput(4)=TrimInput(4);
        goodguess(4)=1;
    else
        TrimInput(4)=TrimInput(4)+ErrEuler(3)*Gain(4);
    end
    Niter=Niter+1;

    if goodguess==[1 1 1 1]
        GoodGuess=1;
    else
        GoodGuess=0;
        exit=input('Hit 1 to Escape Loop, Enter to continue:');
        if exit==1
            GoodGuess=1;
        else
            GoodGuess=0;
        end
    end
end

%Save Initial Guesss
%?Trim condition for lon and lat for constant velocity?
TrimParam.MRCol=TrimInput(3);
TrimParam.TRCol=TrimInput(4);
TrimParam.Lon=TrimInput(1);
TrimParam.Lat=TrimInput(2);
TrimParam.vel=SimStates(end,7:9)';
TrimParam.pos=SimStates(end,1:3)';
TrimParam.euler=SimStates(end,4:6)';
TrimParam.flaps=SimStates(end,13:14)';
TrimParam.pqr=SimStates(end,10:12)';
```

```
clear SimTime SimStates SimOutputs TrimInput

fprintf('\nInitial Guesses for trim Inputs are: Servo Linkage
Command(deg)\n')
fprintf('-----------------------------------   ----------------------
--------\n')
fprintf('     MR Collective Command
%2.3f\n',TrimParam.MRCol*180/pi)
fprintf('     TR Collective Command
%2.3f\n',TrimParam.TRCol*180/pi)
fprintf('Longitudinal Flap Angle Command
%2.3f\n',TrimParam.Lon*180/pi)
fprintf('   Lateral Flap Angle Command
%2.3f\n',TrimParam.Lat*180/pi)

%Perform Helicopter Trim
fprintf('\n');
fprintf('\nPerforming the aircraft trim...\n')
%Set Initial Guesses
InitialStates=zeros(ICParam.NSimulinkStates,1);
InitialStates(1)=TrimParam.pos(1);
InitialStates(2)=TrimParam.pos(2);
InitialStates(3)=TrimParam.pos(3);
InitialStates(4)=TrimParam.euler(1);
InitialStates(5)=TrimParam.euler(2);
InitialStates(6)=TrimParam.euler(3);
InitialStates(7)=TrimParam.vel(1);
InitialStates(8)=TrimParam.vel(2);
InitialStates(9)=TrimParam.vel(3);
InitialStates(10)=TrimParam.pqr(1);
InitialStates(11)=TrimParam.pqr(2);
InitialStates(12)=TrimParam.pqr(3);
InitialStates(13)=TrimParam.flaps(1);
InitialStates(14)=TrimParam.flaps(2);
% Set optimization parameters
TrimParam.Options(1)  = 1;     % show some output
TrimParam.Options(14) = 1000;  % max iterations

InitialInput=[TrimParam.Lon;TrimParam.Lat;TrimParam.MRCol;TrimParam.TRC
ol];
InitialOutput=[TrimParam.pos;TrimParam.vel;TrimParam.euler]
InitialDerivatives=zeros(ICParam.NSimulinkStates,1);
%StateFixIdx is a list of indeces for the states that need to be
%held constant. In this case, [phi theta psi u v w alpha beta p q r]
StateFixIdx=[4 5 6 7 8 9 10 11 12 13 14];
InputFixIdx=[];
OutputFixIdx=[4 5 6 7 8 9];
%State derivatives to be held fixed
DerivFixIdx=[7 8 9 10 11 12 13 14];

%Trim The Helicopter
[TrimOutput.States,TrimOutput.Inputs,TrimOutput.Outputs...
```

```
,TrimOutput.Derivatives,options]=trim(ICParam.SimModel,InitialStates...
    ,InitialInput,InitialOutput,StateFixIdx,InputFixIdx,OutputFixIdx...
    ,InitialDerivatives,DerivFixIdx,TrimParam.Options);
```

APPENDIX E: RUN MODEL FILE

```
%RunModel.m
%This program allows the user to choose desired states, run the model
with
%the chosen states, find the trim conditions for the nonlinear model,
find
%the linear representation, and plot step responses from a central
command
%program
clear all
clf
%Run the configuration file that defines all the model parameters
run Config
run CreateModelStructure
exit=0;
trimed=0;
linear=0;
while exit==0
fprintf('\nMENU\n')
fprintf('You must Choose Option 1 Before any Other Choice\n')
fprintf('=============================================\n')
fprintf('1- Choose Desired States & Trim Heli Model\n')
fprintf('2- Print Trim Results\n')
fprintf('3- Generate Step Responses for NonLinear Model\n')
fprintf('4- Plot Step Responses for Linear Model & Compare\n')
fprintf('5- Display Heave Model Transfer Functions\n')
fprintf('6- Display the Yaw Dynamics Model Transfer Function\n')
fprintf('7- Display the Lon and Lat Linear Models Derived\n')
fprintf('8- Conduct Derivative Analysis\n')
fprintf('9- Exit\n')
menu=input('Enter Your Choice:\n');

    if menu==1
       run TrimHeli
       trimed=1;
       fprintf('NonLinear Model Has Been Trimmed\n')
    elseif menu==2
        if trimed==0
            fprintf('You must First Choose States and Trim, Choose 1 at
the Menu\n')
        elseif trimed==1
            %Print the Trim Results
            fprintf('\n================================\n');
            fprintf('\nThe Trim Results Are:');
            fprintf('\n-----------------------------------------------
-');
            fprintf('\n               MR  Collective  Command  =  %3.3f
(deg)',TrimOutput.Inputs(3)*180/pi);
            fprintf('\n               TR  Collective  Command  =  %3.3f
(deg)',TrimOutput.Inputs(4)*180/pi);
            fprintf('\n   Longitudinal   Cyclic   Command   =   %3.3f
(deg)',TrimOutput.Inputs(1)*180/pi);
```

```
            fprintf('\n              Lateral  Cyclic  Command  =  %3.3f
(deg)\n',TrimOutput.Inputs(2)*180/pi);
        else
            exit=0;
        end
    elseif menu==3
        if trimed==0
            fprintf('You must First Choose States and Trim, Choose 1 at
the Menu\n')
        elseif trimed==1
            clear UT TrimInput
            time=0:(1/SR):runtime;
            samples=runtime*SR+1;
            TrimInput=zeros(4,samples);
            Linear.SimModel='LinearTRexAlign';
            Linear.SimOptions=simget(Linear.SimModel);

            for i=1:4
            %TrimInput(1,:)=TrimParam.Lon;
            %TrimInput(2,:)=TrimParam.Lat;
            %TrimInput(3,:)=TrimParam.MRCol;
            %TrimInput(4,:)=TrimParam.TRCol;
                TrimInput(i,:)=TrimOutput.Inputs(i);
            end
            size=input('Enter desired step size in degrees:\n');
            StepInput(1,101:samples)=pi/180*size;
            LonStep=TrimInput(1,:)'+StepInput';
            LatStep=TrimInput(2,:)'+StepInput';
            MRColStep=TrimInput(3,:)'+StepInput';
            MRColStepDown=TrimInput(3,:)'-StepInput';
            TRColStep=TrimInput(4,:)'+StepInput';
            UT1=[time'    LonStep    TrimInput(2,:)'    TrimInput(3,:)'
TrimInput(4,:)'];
            UT2=[time'    TrimInput(1,:)'    LatStep    TrimInput(3,:)'
TrimInput(4,:)'];
            UT3up=[time'   TrimInput(1,:)'   TrimInput(2,:)'   MRColStep
TrimInput(4,:)'];
            UT3dwn=[time' TrimInput(1,:)' TrimInput(2,:)' MRColStepDown
TrimInput(4,:)'];
            UT4=[time'  TrimInput(1,:)'  TrimInput(2,:)'  TrimInput(3,:)'
TRColStep];
            fprintf('Please wait, running simulation...\n')
            [SimTime,SimStates,SimOutputs1]=sim(ICParam.SimModel,[0
runtime],...
            ICParam.SimOptions,UT1);
            t=SimTime;

            figure(1)
            ymin=min(LonStep)-.25*pi/180;
            ymax=max(LonStep)+.25*pi/180;
            subplot(3,1,1)
            plot(t,UT1(:,2)*180/pi),    grid    on,    axis([0    runtime
ymin*180/pi ymax*180/pi])
            title('Longitudinal Step Input (deg)')
            subplot(3,1,2)
```

```
plot(t,SimOutputs1(:,4),'bo',t,SimOutputs1(:,2),t,SimOutputs1(:,3),'g--
')
            grid on, title('Velocity (m/s)')
            legend('u','v','w','location','northwest');
            subplot(3,1,3)
            plot(t,SimOutputs1(:,8)*180/pi,'r-
',t,SimOutputs1(:,8)*180/pi,'bo')
            grid on, title('Euler Angles (deg)')
            legend('theta','theta linear','location','northwest');
            fprintf('Done Running Longitudinal Step Simulation\n')

            [SimTime,SimStates,SimOutputs2]=sim(ICParam.SimModel,[0
runtime],...
            ICParam.SimOptions,UT2);

            figure(2)
            ymin=(min(LatStep)-.25*pi/180)*180/pi;
            ymax=(max(LatStep)+.25*pi/180)*180/pi;
            subplot(3,1,1)
            plot(t,UT2(:,3)*180/pi),  grid  on,  axis([0  runtime  ymin
ymax])
            title('Lateral Step Input (deg)')
            subplot(3,1,2)

plot(t,SimOutputs2(:,4),t,SimOutputs2(:,5),'bo',t,SimOutputs2(:,6),'g--
')
            grid on, title('Velocity (m/s)')
            legend('u','v','w','location','northwest');
            subplot(3,1,3)
            plot(t,SimOutputs2(:,7)*180/pi,'r-
',t,SimOutputs2(:,8)*180/pi,t,SimOutputs2(:,9)*180/pi,'bo')
            grid on, title('Euler Angles (deg)')
            legend('phi','theta','psi','location','northwest');

            [SimTime,SimStates,SimOutputs3up]=sim(ICParam.SimModel,[0
runtime],...
            ICParam.SimOptions,UT3up);
            %Extract  heave  dynamics  climbing  model  from  data  obtained
from nonlinear
            %model
            cpMR=MRColStep;
            w=SimOutputs3up(:,6);
            heavedata=iddata(w,cpMR,1/SR);
            heavemodel_climb=n4sid(heavedata,3);
            [e,xoMRc]=pe(heavemodel_climb,heavedata);
            fprintf('The Climb Heave Model is:\n')
            [ahc,bhc,chc,dhc]=ssdata(heavemodel_climb);
            [numh,denh]=ss2tf(ahc,bhc,chc,dhc);
            hc=tf(numh,denh);

            figure(3)
            ymin=(min(MRColStep)-.25*pi/180)*180/pi;
            ymax=(max(MRColStep)+.25*pi/180)*180/pi;
            subplot(2,1,1)
```

```
            plot(t,UT3up(:,4)*180/pi), grid on, axis([0 runtime ymin
ymax])
            title('MR Step Input (deg)')
            subplot(2,1,2)
            plot(t,SimOutputs3up(:,6),'bo')
            grid on, title('Velocity (m/s)')
            legend('w','location','northwest');

            [SimTime,SimStates,SimOutputs3dwn]=sim(ICParam.SimModel,[0
runtime],...
            ICParam.SimOptions,UT3dwn);

            %Extract heave dynamics descent model from data obtained
from nonlinear
            %model
            cpMRd=MRColStepDown;
            wd=SimOutputs3dwn(:,6);
            heavedatad=iddata(wd,cpMRd,1/SR);
            heavemodel_descent=n4sid(heavedatad,3);
            [e,xoMRd]=pe(heavemodel_descent,heavedatad);
            fprintf('The Descent Heave Model is:\n')
            [ahd,bhd,chd,dhd]=ssdata(heavemodel_descent);
            [numd,dend]=ss2tf(ahd,bhd,chd,dhd);
            hd=tf(numd,dend);

            figure(4)
            ymin=(min(MRColStepDown)-.25*pi/180)*180/pi;
            ymax=(max(MRColStepDown)+.25*pi/180)*180/pi;
            subplot(2,1,1)
            plot(t,UT3dwn(:,4)*180/pi), grid on, axis([0 runtime ymin
ymax])
            title('MR Step Input (deg)')
            subplot(2,1,2)
            plot(t,SimOutputs3dwn(:,6),'bo')
            grid on, title('Velocity (m/s)')
            legend('w','location','northwest');

            [SimTime,SimStates,SimOutputs4]=sim(ICParam.SimModel,[0
runtime],...
            ICParam.SimOptions,UT4);

            cpt=UT4(:,5);  %tail rotor input data
            psi=SimOutputs4(:,9);
            yawdata=iddata(psi,cpt,1/SR); %create the IDDATA object
            yawmodel=n4sid(yawdata);       %generate the identification
model
            [e,xot]=pe(yawmodel,yawdata);  %obtain error and initial
states
            fprintf('The Yaw Dynamics TF is:\n')
            [ay,by,cy,dy]=ssdata(yawmodel);
            [numy,deny]=ss2tf(ay,by,cy,dy);
            yaw=tf(numy,deny);

            figure(5)
```

```
            ymin=(min(TRColStep)-.25*pi/180)*180/pi;
            ymax=(max(TRColStep)+.25*pi/180)*180/pi;
            subplot(2,1,1)
            plot(t,UT4(:,5)*180/pi),  grid  on,  axis([0  runtime  ymin
ymax])
            title('TR Step Input (deg)')
            subplot(2,1,2)
            plot(t,SimOutputs4(:,9)*180/pi,'bo')
            grid on, title('Euler Angles (deg)')
            legend('psi','location','northwest');

        end
    elseif menu==4
        fprintf('Note  that  the  model  that  will  be  extracted  uses  the
current trim settings.\n')
        %Extract the linear model
        fprintf('\n \nExtracting Helicopter Linear Model...\n');
        %Perturbation Level
        LinParam(1)=10^-8;
        [A,B,C,D]=linmod(ICParam.SimModel,TrimOutput.States,...
        TrimOutput.Inputs,LinParam);
        ssplant=ss(A,B,C,D)
        run ExtractSISO
        linear=1;
        fprintf('Lat  and  Long  Model  Extracted  at  chosen  operation
point\n')

        [SimTime,LinStates,LinOutputs1]=sim(Linear.SimModel,[0
runtime],...
                ICParam.SimOptions,UT1);
        figure(1)
        ymin=(min(LonStep)-.25*pi/180)*180/pi;
        ymax=(max(LonStep)+.25*pi/180)*180/pi;
        subplot(3,1,2)

plot(t,SimOutputs1(:,4),'bo',t,SimOutputs1(:,5),t,SimOutputs1(:,6),'g--
',t,LinOutputs1(:,4),'g.')
        legend('u','v','w','u linear','location','northwest');
        grid on
        title('Velocity (m/s)')
        subplot(3,1,3)

plot(t,SimOutputs1(:,8)*180/pi,'bo',t,LinOutputs1(:,8)*180/pi,'g.')
        grid on, title('Euler Angles (deg)')
        legend('theta','theta linear','location','northwest');

        [SimTime,LinStates,LinOutputs2]=sim(Linear.SimModel,[0
runtime],...
                ICParam.SimOptions,UT2);

        figure(2)
        subplot(4,1,2)

plot(t,SimOutputs2(:,4),'b.',t,SimOutputs2(:,5),'bo',t,SimOutputs2(:,6)
,'r--',t,LinOutputs2(:,5),'g.')
```

```
        grid on, title('Velocity (m/s)')
        legend('u','v','w','v linear','location','northwest');
        subplot(4,1,3)

plot(t,SimOutputs2(:,7)*180/pi,'bo',t,LinOutputs2(:,7)*180/pi,'g.')
        grid on, title('Euler Angles (deg)')
        legend('phi','phi linear','location','northwest');
        subplot(4,1,4)
        plot(t,SimOutputs2(:,9)*180/pi)

        [SimTime,LinStates,LinOutputs3up]=sim(Linear.SimModel,[0
runtime],...
                ICParam.SimOptions,UT3up);

        %Extract heave dynamics climbing model from data obtained from
nonlinear
        %model
        cpMR=UT3up(:,4);
        heavedata=iddata(w,cpMR,1/SR);
        heavemodel_climb=n4sid(heavedata,3);
        [e,xoMRc]=pe(heavemodel_climb,heavedata);
        fprintf('The Climb Heave Model is:\n')
        [ahc,bhc,chc,dhc]=ssdata(heavemodel_climb);
        [numh,denh]=ss2tf(ahc,bhc,chc,dhc);
        hc=tf(numh,denh)

        figure(3)
        subplot(2,1,2)
        plot(t,SimOutputs3up(:,6),'bo',t,LinOutputs3up(:,6),'g.')
        grid on, title('Velocity (m/s)')
        legend('w','w linear','location','northwest');

        [SimTime,LinStates,LinOutputs3dwn]=sim(Linear.SimModel,[0
runtime],...TrimInput=[LonCyc   LatCyc   MRCol   TRCol]*TrimInput=[LonCyc
LatCyc MRCol TRCol]*pi/180;pi/180;
                ICParam.SimOptions,UT3dwn);

        cpMR=UT3dwn(:,4);
        heavedata=iddata(SimOutputs3dwn(:,6),cpMR,1/SR);
        heavemodel_descent=n4sid(heavedata,3);
        [e,xoMRd]=pe(heavemodel_descent,heavedata);
        fprintf('The Descent Heave Model is:\n')
        [ahd,bhd,chd,dhd]=ssdata(heavemodel_descent);
        [numd,dend]=ss2tf(ahd,bhd,chd,dhd);
        hd=tf(numd,dend)

        figure(4)
        subplot(2,1,2)
        plot(t,SimOutputs3dwn(:,6),'bo',t,LinOutputs3dwn(:,6),'g.')
        grid on, title('Velocity (m/s)')
        legend('w','w linear','location','southwest');

        [SimTime,LinStates,LinOutputs4]=sim(Linear.SimModel,[0
runtime],...
```

```
                ICParam.SimOptions,UT4);

        %extract linear model for yaw dynamics using SysID toolbox
        %Extract the Yaw dynamics model from data obtained from
nonlinear
        %model
        cpt=UT4(:,5);                   %tail rotor input data
        yawdata=iddata(psi,cpt,1/SR); %create the IDDATA object
        yawmodel=n4sid(yawdata);          %generate the identification
model
        [e,xot]=pe(yawmodel,yawdata); %obtain error and initial states
        fprintf('The Yaw Dynamics TF is:\n')
        [ay,by,cy,dy]=ssdata(yawmodel);
        [numy,deny]=ss2tf(ay,by,cy,dy);
        yaw=tf(numy,deny)

        figure(5)
        subplot(2,1,2)

plot(t,SimOutputs4(:,9)*180/pi,'bo',t,LinOutputs4(:,9)*180/pi,'g.')
        grid on, title('Euler Angles (deg)')
        legend('psi','psi linear','location','southwest');
    elseif menu==5
        fprintf('The Climbing Heave Model is:\n')
        hc
        fprintf('The Descent Heave Model is:\n')
        hd
    elseif menu==6
        fprintf('The Yaw Dynamics Model is:\n')
        yaw
    elseif menu==7
        fprintf('The Longitudinal Models are:\n')
        fprintf('dlon to x\n')
        lon_x_tf
        fprintf('dlon to u\n')
        lon_u_tf
        fprintf('dlon to theta\n')
        lon_theta_tf
        pause
        fprintf('The Lateral Models are:\n')
        fprintf('dlat to y\n')
        lat_y_tf
        fprintf('dlat to v\n')
        lat_v_tf
        fprintf('dlat to phi\n')
        lat_phi_tf
    elseif menu==8
        %Resize Matrix A to fit Padfield pg. 210
        %go from state x=[x y z phi theta psi u v w p q r alpha beta]
to
        % x=[u w q theta v p phi r]
        dummyA=zeros(8);
        dummyB=zeros(8,4);
        column=[7 9 11 5 8 10 4 12];
        count=0;
```

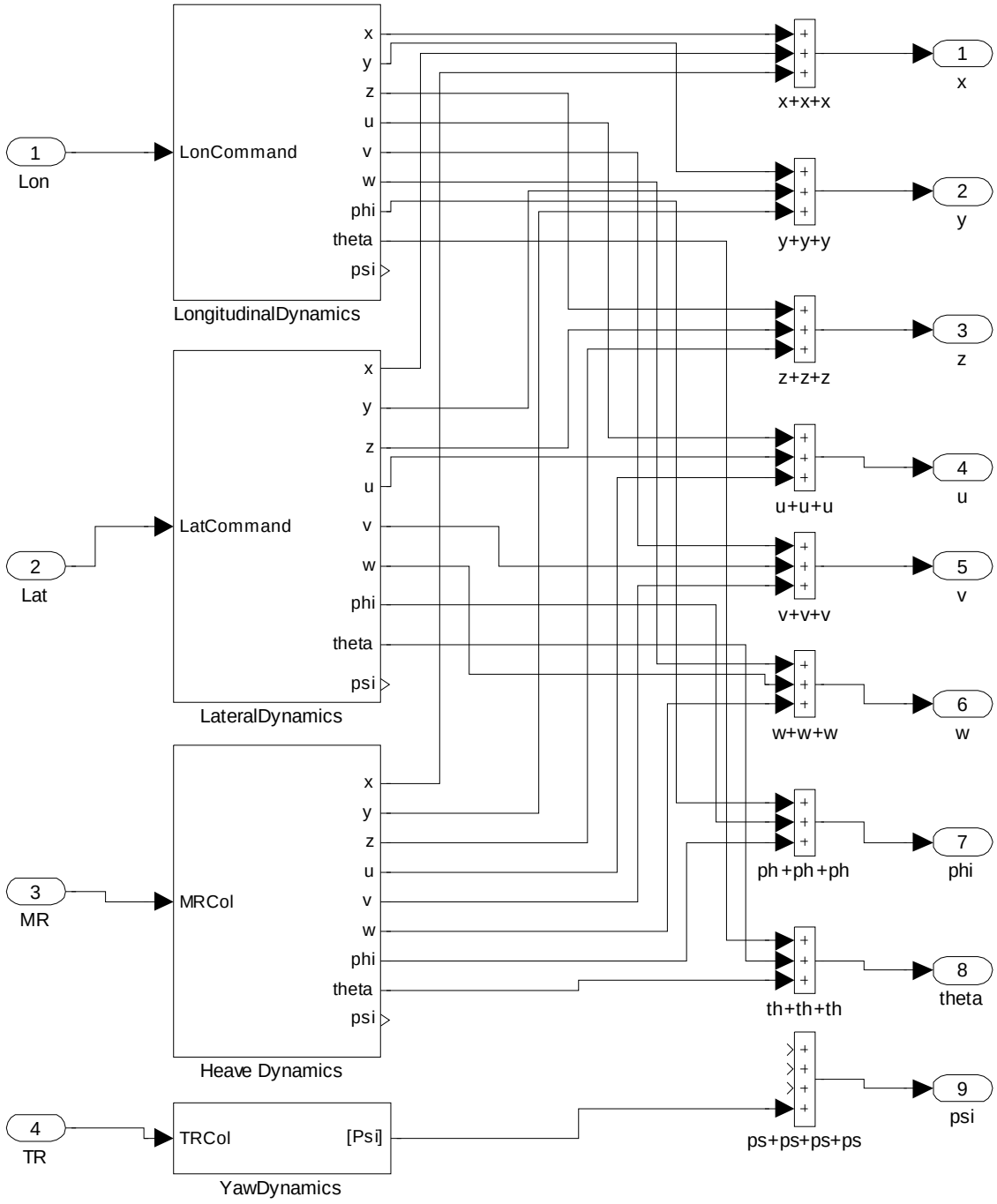
```
        for j=1:8
        count=count+1;
            for i=1:8
                col=column(i);
                dummyA(j,i)=A(column(j),col);
                if count==1
                dummyB(i,:)=B(col,:);
                else
                end
            end

        end
        Apadfield=dummyA
        Bpadfield=dummyB

    elseif menu==9
        exit=1;
    end
end
```

APPENDIX F: EXTRACT SINGLE INPUT SINGLE OUTPUT MODELS

```
%Extract SISO Models
%Make sure you already have the linear plant model defined
%The model ssplant is defined in RunModel.m, menu option 3

%Define the linear longitudinal model: input=dlon  output=[x u theta]
%Model Inputs=[dlon dlat MRCol TRCol]
%              1 2 3 4 5 6  7    8    9
%ModelOutputs=[x y z u v w phi theta psi]

%=======================================
%Longitudinal Model
%=======================================
lon_to_x=minreal(ssplant(1,1));
lon_to_y=minreal(ssplant(2,1));
lon_to_z=minreal(ssplant(3,1));
lon_to_u=minreal(ssplant(4,1));
lon_to_v=minreal(ssplant(5,1));
lon_to_w=minreal(ssplant(6,1));
lon_to_phi=minreal(ssplant(7,1));
lon_to_theta=minreal(ssplant(8,1));
lon_to_psi=minreal(ssplant(9,1));

%======================================
%Lateral Model
%======================================
lat_to_x=minreal(ssplant(1,2));
lat_to_y=minreal(ssplant(2,2));
lat_to_z=minreal(ssplant(3,2));
lat_to_u=minreal(ssplant(4,2));
lat_to_v=minreal(ssplant(5,2));
lat_to_w=minreal(ssplant(6,2));
lat_to_phi=minreal(ssplant(7,2));
lat_to_theta=minreal(ssplant(8,2));
lat_to_psi=minreal(ssplant(9,2));

%======================================
%Heave Dynamics Model
%======================================
MRCol_to_x=minreal(ssplant(1,3));
MRCol_to_y=minreal(ssplant(2,3));
%MRCol_to_z=minreal(ssplant(3,3));
MRCol_to_u=minreal(ssplant(4,3));
MRCol_to_v=minreal(ssplant(5,3));
%MRCol_to_w=minreal(ssplant(6,3));
MRCol_to_phi=minreal(ssplant(7,3));
MRCol_to_theta=minreal(ssplant(8,3));
MRCol_to_psi=minreal(ssplant(9,3));

%====================================
```

```
%Yaw Dynamics Model
%====================================
TRCol_to_x=minreal(ssplant(1,4));
TRCol_to_y=minreal(ssplant(2,4));
TRCol_to_z=minreal(ssplant(3,4));
TRCol_to_u=minreal(ssplant(4,4));
TRCol_to_v=minreal(ssplant(5,4));
TRCol_to_w=minreal(ssplant(6,4));
TRCol_to_phi=minreal(ssplant(7,4));
TRCol_to_theta=minreal(ssplant(8,4));
%TRCol_to_psi=minreal(ssplant(9,4));
```

APPENDIX G: LINEAR T-REX ALIGN 600 MODEL STRUCTURE



THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX H: MINIMUM REALIZATION TRANSFER FUNCTION MODELS AT HOVER (1° STEP COMMAND)

Longitudinal Dynamics Transfer Functions

$$\frac{x}{\beta_{lon}} = \frac{2.842\times10^{-14}s^5 + 6.821\times10^{-13}s^4 + 98.05s^3 + 3369s^2 + 2.405\times10^4 s + 1633}{s^6 + 44.39s^5 + 590.3s^4 + 2488s^3 + 243.5s^2 + 5.189s - 3.628\times10^{-12}} \text{ (m/rad)}$$

$$\frac{u}{\beta_{lon}} = \frac{2.842\times10^{-14}s^4 + 98.05s^3 + 3369s^2 + 2.405\times10^4 s + 1633}{s^5 + 44.39s^4 + 590.3s^3 + 2488s^2 + 243.5s + 5.189} \quad \text{(m/sec-rad)}$$

$$\frac{\theta}{\beta_{lon}} = \frac{3.553\times10^{-14}s^4 + 7.958\times10^{-13}s^3 - 0.06428s^2 - 0.6428s + 1.231\times10^{-9}}{s^5 + 44.29s^4 + 585.9s^3 + 2429s^2 + 0.0003249s + 0.001347} \text{ (rad/rad)}$$

Lateral Dynamics Transfer Functions

$$\frac{y}{\beta_{lat}} = \frac{-5.4\times10^{-13}s^5 - 7.071\times10^{-11}s^4 + 98.05s^3 + 1.003\times10^4 s^2 + 9.074\times10^4 s + 2817}{s^6 + 112.3s^5 + 1956s^4 + 9416s^3 + 923.6s^2 + 19.7s + 6.944\times10^{-13}}$$

(m/rad)

$$\frac{v}{\beta_{lat}} = \frac{-1.421\times10^{-13}s^4 + 98.05s^3 + 1.003\times10^4 s^2 + 9.074\times10^4 s + 2817}{s^5 + 112.3s^4 + 1956s^3 + 9416s^2 + 923.6s + 19.7} \text{ (m/sec-rad)}$$

$$\frac{\phi}{\beta_{lat}} = \frac{2.842\times10^{-14}s^4 - 4.547\times10^{-13}s^3 + 0.1214s^2 + 1.214s + 1.671\times10^{-8}}{s^5 + 112.2s^4 + 1944s^3 + 9222s^2 + 0.001078s + 0.005114} \text{ (rad/rad)}$$

Climbing Heave Dynamics Transfer Functions

$$\frac{w}{\theta_o} = \frac{2.061s^2 - 2.73s + .6697}{s^3 - 1.985s^2 + .9854s - 5.7\times10^{-4}} \text{ (m/sec-rad)}$$

Descending Heave Dynamics Transfer Functions

$$\frac{w}{\theta_o} = \frac{2.064s^2 - 3.983s - 1.919}{s^3 - 2.931s^2 + 2.862s - 0.9317} \text{ (m/sec-rad)}$$

Yaw Dynamics Transfer Function

$$\frac{\psi}{\delta} = \frac{1.691\times10^{-7}s^5 - 0.0128s^4 + 0.0161s^3 + 0.008953s^2 - 0.0161s + 0.00383}{s^6 - 3.668s^5 + 4.62s^4 - 1.71s^3 - 0.9366s^2 + 0.844s - 0.1545} \text{ (rad/rad)}$$

MODE	**EIGENVALUES**
Longitudinal Velocity (u)	$\begin{bmatrix} -24.3 & -10.14 & -9.86 & -.0686 & -.0312 \end{bmatrix}$
Pitch	$\begin{bmatrix} -24.3 & -10.3 & -9.7 & \pm 7\times10^{4}\, j \end{bmatrix}$
Lateral Velocity (v)	$\begin{bmatrix} -92.19 & -10.3 & -9.7 & -.0685 & -.0312 \end{bmatrix}$
Roll	$\begin{bmatrix} -92.2 & -10 \pm .156\, j & \pm 7.44\times10^{-4}\, j \end{bmatrix}$

LIST OF REFERENCES

[1] Vladislav Gavrilets, “Autonomous Aerobatic Maneuvering of Miniature Helicoptersrotary-wing," PhD Thesis, MIT, Boston, MA, 2003.

[2] Bernard Mettler, *Identification Modeling and Characteristics of Miniature Rotorcraft*, Kluwer Academic Publishers, 2003.

[3] Raymond W. Prouty, *Helicopter Performance, Stability, and Control*, PWS Publishers, 1986.

[4] Gareth D. Padfield, *Helicopter Flight Dynamics: The Theory and Application of Flying Qualities and Simulation Modeling*, AIAA Education Series, 1996.

[5] J. Gordon Leishman, *Principles of Helicopter Aerodynamics*, Cambridge University Press, 2000.

[6] John D. Anderson, Jr., *Fundamentals of Aerodynamics*, McGraw-Hill, Inc., 1991.

[7] Jim Ledin, *Embedded Control Systems in C/C++*, CMP Books, 2004.

[8] Katsuhiko Ogata, *Modern Control Engineering*, Pearson Prentice Hall, 2003.

[9] Katsuhiko Ogata, *MATLAB for Control Engineers*, Pearson Prentice Hall, 2008.

[10] Norman S. Nise, *Control Systems Engineering*, John Wiley & Sons, Inc., 2004.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, VA

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, CA

3. Dr. Jeffrey Knorr
   Electrical and Computer Engineering Department
   Code EC/Ko
   Naval Postgraduate School
   Monterey, CA

4. Dr. Robert G. Hutchins
   Electrical and Computer Engineering Department
   Naval Postgraduate School
   Monterey, CA

5. Dr. Vladimir Dobrokhodov
   Mechanical Engineering Department
   Naval Postgraduate School
   Monterey, CA

6. Ioannis Kitsios
   Mechanical Engineering Department
   Naval Postgraduate School
   Monterey, CA

7. Marine Corps Representative
   Naval Postgraduate School
   Monterey, CA

8. Director, Training and Education, MCCDC, Code C46
   Quantico, VA

9. Director, Marine Corps Research Center, MCCDC, Code C40RC
   Quantico, VA

10. Marine Corps Tactical Systems Support Activity (Attn: Operations Officer)
    Camp Pendleton, CA